UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INFORMÁTICA SUPERIOR


PROYECTO FIN DE CARRERA

# PERIFÉRICO DE MONITORIZACIÓN Y CONTROL PARA SISTEMAS EMPOTRADOS Y SERVIDORES

AUTOR: MIGUEL PERIS FERNÁNDEZ
TUTOR: ÓSCAR PÉREZ ALONSO

17 de junio de 2013

*SDUD PRQWVH, SRU VHU OR PHMRU TXH PH KD SDVDGR HQ OD YLGD.*
*WH TXLHUR.*

# Agradecimientos

Tras un año de trabajo el trabajo este proyecto este proyecto va a ver la luz. Quiero agradecer su ayuda y apoyo a mis padres y mi hermano Luis. A mi amiga Zaira que sin ella no hubiera terminado la carrera porque lo hubiera dejado todo para último momento. Y a mi tutor Óscar, por permitirme hacer un proyecto que me apasiona.

# Resumen

Existen gran cantidad de equipos informáticos en los que no se dispone de periféricos de entrada/salida estándar como un monitor ni un teclado o ratón. Dentro de estos se pueden encontrar servidores de cualquier tipo y sistemas empotrados desempeñando una tarea específica, que no pueden disponer de estos periféricos por razones de espacio u otras razones justificadas.

En ocasiones es necesario obtener información de estos equipos, como consultar su estado (memoria, carga, etc.) y poder realizar tareas simples tales como reiniciar o apagar la máquina de una manera adecuada y segura.

Para solventar estas cuestiones en el presente proyecto de fin de carrera se desarrolla un periférico de monitorización y control que permitirá monitorizar distintos factores relevantes así como realizar tareas de control simples como apagar y reiniciar la máquina. Este periférico se conectar a la máquina por el puerto USB suministrando alimentación al mismo y de un medio de comunicación con el equipo. Además, dispondrá de unos sensores de temperatura que permitirán obtener la temperatura ambiente de la estancia donde se encuentre el periférico. El proyecto abarca el desarrollo del hardware así como del software necesario.

**Palabras clave:** periférico, monitorización, control, microcontrolador, AVR, USB, sistema empotrado, hardware, software.

# Abstract

There are many computers that are not available standard input/output peripherals like a monitor, keyboard or mouse. Within this range can find any kind of servers or embedded systems to play a specific task, can not avail this peripherals for reasons of space or other justifiable reasons.

Sometimes is necessary get information of this machines like get their status (memory, load, etc.) and perform simple tasks such as shutdown or reboot the machine properly and safely.

To solve this issues in this project will be developed a peripheral of monitoring and control which will monitoring some relevant factors also perform simple control tasks like shutdown and reboot the machine. This peripheral will connect to the computer through USB port which will provide the main power and communication with the computer. Also have temperature sensors that permit get the ambient temperature of the room where the peripheral is. The project includes the development of hardware and the software required.

**Keywords:** peripheral, monitoring, control, microcontroller, AVR, USB, embedded system, hardware, software.

# Índice

**1. Introducción** **7**
1.1. Motivación . . . 7
1.2. Objetivos . . . 8
1.3. Contenido de la memoria . . . 8

**2. Estado del arte** **10**
2.1. Introducción a los microcontroladores . . . 10
2.2. GNU/Linux . . . 13
2.3. Protocolo USB . . . 13
2.4. Lenguaje C . . . 14
2.5. Componentes electrónico básicos . . . 14
2.6. Redundancia de Hardware . . . 18

**3. Análisis del sistema** **19**
3.1. Análisis de la plataforma de trabajo . . . 19
3.1.1. Análisis de la plataforma Hardware . . . 19
3.1.2. Análisis de la plataforma Software . . . 21
3.2. Descripción General . . . 22
3.2.1. Capacidades generales . . . 22
3.2.2. Restricciones generales . . . 23
3.3. Requisitos de Usuario . . . 24
3.3.1. Requisitos de Capacidad . . . 24
3.3.2. Requisitos de Restricción . . . 27
3.3.3. Requisitos Hardware . . . 28

**4. Diseño del sistema** **30**
4.1. Diseño Hardware . . . 30
4.1.1. Diseño del circuito . . . 30
4.1.2. Diseño de la PCB . . . 34
4.2. Diseño Software . . . 38
4.2.1. Requisitos Software . . . 38
4.2.2. Diseño del firmware . . . 42
4.2.3. Diseño del demonio . . . 46

**5. Implementación** **49**
5.1. Implementación hardware . . . 49
5.1.1. Prototipo . . . 49
5.1.2. Montaje de la placa de circuito impreso . . . 53
5.2. Implementación del firmware . . . 59
5.3. Implementación del demonio . . . 65

**6. Pruebas** **69**
6.1. Pruebas unitarias . . . 69
6.2. Pruebas integradas . . . 76
6.3. Pruebas de consumo . . . 84

**7. Consideraciones finales** **88**
7.1. Planificación . . . . . 88
7.2. Presupuesto . . . . . 90
7.3. Conclusiones . . . . . 93
7.4. Trabajos futuros . . . . . 94

**8. Acrónimos y definiciones** **97**

**Bibliografía** **99**

**A. Anexo 1: Compilación del código y programación del microcontrolador** **100**

# Índice de figuras

1. Estructura interna de un microcontrolador . . . 10
2. Logotipo de Microchip . . . 11
3. Logotipo de Atmel . . . 12
4. Tipos de conectores USB . . . 14
5. Resistencia . . . 15
6. Condensador electrolítico . . . 16
7. Condensador cerámico . . . 16
8. Condensador de película . . . 16
9. Circuito integrado . . . 17
10. Ejemplo de Triple Modular Redundancy . . . 18
11. Cuota de mercado SO en servidores (a fecha de 27/04/2013) . . . 21
12. Cuota SO Unix en servidores (a fecha de 27/04/2013) . . . 22
13. Diagrama de bloques del circuito . . . 30
14. Microcontrolador AVR . . . 31
15. Pantalla LCD . . . 31
16. Sensores de temperatura . . . 32
17. Pulsadores de control . . . 32
18. USB . . . 33
19. Circuito completo . . . 34
20. Boceto de la PCB (medidas en cm.) . . . 35
21. Diseño de la PCB . . . 36
22. PCB fabricada (cara superior) . . . 37
23. PCB fabricada (cara inferior) . . . 37
24. Diagrama de flujo . . . 43
25. Diagrama de flujo del demonio . . . 47
26. Prototipo del periférico . . . 49
27. Prototipo: Pantalla LCD . . . 50
28. Prototipo: Microcontrolador . . . 50
29. Prototipo: Sensores de temperatura . . . 51
30. Prototipo: USB . . . 51
31. Prototipo: Pulsadores . . . 52
32. Herramientas necesarias para el montaje . . . 53
33. Componentes del periférico . . . 54
34. Montaje del periférico 1 . . . 55
35. Montaje del periférico 2 . . . 55
36. Montaje del periférico 3 . . . 56
37. Montaje del periférico 4 . . . 56
38. Montaje del periférico 5 . . . 57
39. Montaje del periférico 6 . . . 58
40. Periférico finalizado . . . 58
41. Distribución en pantalla . . . 60
42. Menús implementados: Temperatura . . . 62
43. Menús implementados: Monitorización . . . 63
44. Menús implementados: Control . . . 64
45. Menús implementados: Acerca . . . 64
46. Temperatura de los tres sensores . . . 70
47. Lectura de la temperatura de los tres sensores . . . 71

48. Resultados de los algoritmos de votación . . . . . 72
49. Fallo de un sensor . . . . . 73
50. Conexión de nuevo del sensor . . . . . 74
51. Fallo de dos sensores . . . . . 75
52. Raspberry Pi . . . . . 76
53. Elementos hardware necesarios . . . . . 77
54. Conexión de los elementos . . . . . 78
55. Prueba: Fecha y hora del sistema . . . . . 79
56. Prueba: Espacio en disco . . . . . 79
57. Prueba: Uptime y carga del sistema . . . . . 80
58. Prueba: Memoria RAM libre . . . . . 80
59. Prueba: Kernel . . . . . 81
60. Prueba: Información de Red . . . . . 81
61. Prueba: Temperatura de la CPU . . . . . 82
62. Prueba: Reiniciar equipo . . . . . 82
63. Prueba: Apagar equipo . . . . . 83
64. Latiguillo USB . . . . . 85
65. Latiguillo USB . . . . . 86
66. Planificación . . . . . 89
67. Pantalla LCD 128x64 . . . . . 94
68. Módulo bluetooth . . . . . 95
69. Proyecto Sistemas Informáticos . . . . . 96
70. Programador USBasp . . . . . 102
71. Conexión del periférico con el programador . . . . . 103

# Índice de tablas


1. Comparativa microcontroladores posibles . . . 20
2. Plantilla de requisitos . . . 24
3. RUC-01 . . . 24
4. RUC-02 . . . 25
5. RUC-03 . . . 25
6. RUC-04 . . . 25
7. RUC-05 . . . 25
8. RUC-06 . . . 26
9. RUC-07 . . . 26
10. RUC-08 . . . 26
11. RUC-09 . . . 26
12. RUR-01 . . . 27
13. RUR-02 . . . 27
14. RUR-03 . . . 27
15. RUR-04 . . . 27
16. RUR-05 . . . 28
17. RUR-06 . . . 28
18. RHW-01 . . . 28
19. RHW-02 . . . 28
20. RHW-03 . . . 29
21. RHW-04 . . . 29
22. RHW-05 . . . 29
23. RSF-01 . . . 38
24. RSF-02 . . . 38
25. RSF-03 . . . 38
26. RSF-04 . . . 39
27. RSF-05 . . . 39
28. RSF-06 . . . 39
29. RSF-07 . . . 39
30. RSF-08 . . . 40
31. RSI-01 . . . 40
32. RSI-02 . . . 40
33. RSI-03 . . . 41
34. RSR-01 . . . 41
35. RSO-01 . . . 41
36. RSO-02 . . . 42
37. Comandos enviados desde el dispositivo . . . 64
38. Comandos de control enviados desde el dispositivo . . . 65
39. Costes de Personal . . . 90
40. Costes de Equipo . . . 90
41. Costes de Hardware . . . 91
42. Otros costes . . . 91
43. Presupuesto final . . . 92

# 1. Introducción

En la actualidad existen multitud de máquinas y equipos que no disponen de un monitor conectado la mayor parte del tiempo, como por ejemplo un servidor o un sistema empotrado, y es necesario tener el equipo conectado a la red y disponer de un ordenador para poder acceder a la máquina. Esto en algunos casos no es posible, como en algunos sistemas empotrados no conectados a la red, o se desea tener un sistema de monitorización in situ, sin necesidad de tener una máquina adicional. Esto es lo que se intenta resolver con el presente proyecto fin de carrera.

El proyecto consiste en la creación de un **periférico** conectado por **USB** a un ordenador con sistema operativo Linux que permitirá la **monitorización** y ejecución de **tareas simples** sin la necesidad de un monitor. El periférico estará gobernado por un microcontrolador de **Atmel** de 8 bits y dispondrá de una **pantalla LCD** donde se mostrará la información.

El proyecto se centrará tanto en el **desarrollo hardware** del dispositivo, como en el **desarrollo software** del **microcontrolador**, así como la implementación de un **demonio** para Linux.

## 1.1. Motivación

La motivación de este proyecto es por un lado el diseño de un circuito electrónico basado en un microcontrolador de **Atmel**, incluyendo el desarrollo del firmware del mismo así como el diseño de una Placa de Circuito Integrado (PCB) y por otro lado investigar sobre la comunicación **USB** entre el dispositivo hardware creado y el PC, incluyendo un demonio para poder interactuar con el dispositivo.

El dispositivo estaría indicado para la **monitorización de sistemas** en los que se carezca de pantalla y periféricos de entrada (ratón y teclado), como por ejemplo un **sistema empotrado** (en este caso una Raspberry Pi) o un servidor metido en un rack, o simplemente se desea acceso rápido a la información.

El periférico estaría compuesto por una **pantalla LCD** y unos **pulsadores**, gobernado por un **microcontrolador de 8 bits** de Atmel, que realizaría las tareas de gestión de la pantalla y los menús, además de la comunicación vía **USB** con el equipo. Por otro lado, el periférico contaría con varios **sensores de temperatura**, que medirán la temperatura ambiente y se presentará en el dispositivo según varios algoritmos de votación.

Las tareas principales que realizaría este dispositivo serían las siguientes:

- **Visualización** de datos enviados por el PC en la pantalla del dispositivo.
- Visualizar la **temperatura ambiente** leída de los sensores del dispositivo.
- Enviar datos desde **el dispositivo al PC** (la temperatura leída por los sensores).
- Manejo del dispositivo con una serie de **menús**.

Por el lado del software, el **demonio** será desarrollado para un sistema **GNU/Linux** genérico en el lenguaje de programación C y el firmware del microcontrolador será desarrollado igualmente en C, apoyándose en las librerías AVR-LIBC.

## 1.2. Objetivos

Los objetivos del trabajo dirigido serán los siguientes:

- **Diseño y montaje de un prototipo.**
  Se diseñara el circuito con su diagrama esquemático para posteriormente realizar un prototipo del mismo.
- **Diseño y fabricación de PCB.**
  Partiendo del circuito se diseñará una placa de circuito integrado (PCB) y se mandará a una empresa especializada para su fabricación.
- **Realizar una comunicación bidireccional por USB, entre el dispositivo el ordenador.**
  El dispositivo podrá comunicarse con el ordenador y viceversa.
- **Crear un dispositivo intuitivo**
  El dispositivo será intuitivo y podrá ser manejado con una serie de menús sencillos.
- **Algoritmos de votación de temperatura**
  Se implementarán varios algoritmos de votación para presentar una sola medida de temperatura a partir de los 3 sensores y poseer características de un sistema tolerante a fallos.
- **Programación de un demonio para Linux.**
  Se desarrollará un demonio para poder enviar/recibir información hacia/desde el dispositivo.

## 1.3. Contenido de la memoria

En esta sección se explicará el contenido de este documento, detallando brevemente cada uno de los capítulos que lo componen. Los capítulos que componen la memoria son los siguientes:

1. **Introducción**
   En este apartado se comentará la motivación del proyecto además de los objetivos del mismo. Se trata del capítulo actual.
2. **Estado del arte**
   Este capítulo se mostrarán las tecnologías utilizadas en el proyecto.
3. **Análisis**
   Aquí se procederá a definir las especificaciones y requisitos del proyecto.
4. **Diseño**
   En este apartado se muestran las soluciones propuesta a los requisitos especificados en el apartado anterior, además de como actuar en el desarrollo del proyecto. Se realizará el diseño tanto del hardware como del software.
5. **Implementación**
   Aquí se procederá a describir la implementación del hardware y del software así como la integración entre ambos.
6. **Pruebas**
   Se procederá a realizar pruebas unitarias e integradas del software y hardware desarrollado, así como unas pruebas de consumo.

7. **Consideraciones finales**
   En esta sección se definirá la planificación del proyecto, así como el presupuesto del mismo. Por otro lado, se expondrán las conclusiones del proyecto y los trabajos futuros.

8. **Anexo 1: Compilación del código y programación del microcontrolador**
   En este anexo se explicará como compilar todo el código fuente generado en le proyecto además de como grabar el firmware del microcontrolador.

# 2. Estado del arte

A continuación se procede a comentar brevemente el estado del arte de este trabajo. En este caso se hará una pequeña introducción a que son los microntroladores, además de comentar que es el protocolo USB y algunos componentes electrónicos básico. Por el lado del software, se hablará del sistema operativo GNU/Linux y del lenguaje de programación C.

## 2.1. Introducción a los microcontroladores

Un microcontrolador es un circuito integrado que contiene en el mismo encapsulado las tres unidades funcionales básicas de un ordenador, a saber, **unidad central de procesamiento (CPU)**, **memoria** y **periféricos de entrada/salida**.

Se fabrican para reducir el coste económico y de consumo de energía de un sistema particular, acotando las características del mismo a las necesidades del sistema donde se va a utilizar.

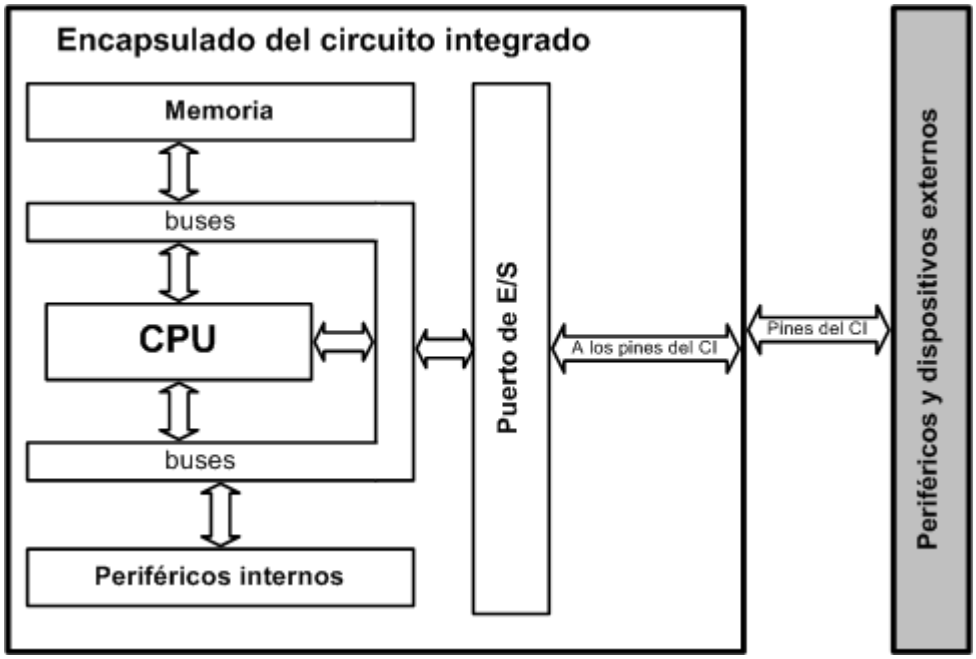


Figura 1: Estructura interna de un microcontrolador

Pese que en la actualidad existen multitud de fabricantes de microcontroladores en el mercado, en el ámbito educativo y aficionado (aunque también en otros ámbitos) los microcontroladores más utilizados son los **PIC** de la empresa Microchip y los **AVR** de Atmel.

A continuación se van describir brevemente las dos familias de microcontroladores:

**Microcontroladores PIC**


Figura 2: Logotipo de Microchip

Los microcontroladores PIC son una familia de microcontroladores RISC fabricados por Microchip basados en el PIC1650 desarrollado originalmente por General Instruments.

Las características principales de los microcontroladores PIC son las siguientes:

- Arquitectura Harvard (el área de código y datos están separadas).
- Un número de instrucciones reducido y de longitud fija.
- La mayoría de las instrucciones se ejecutan en un ciclo de reloj(2 o 4 ciclos para los modelos de 8 bits), con un ciclo extra para bifurcaciones y saltos.
- Un solo acumulador (W), cuyo uso (como operador de origen) es implícito (no está especificado en la instrucción).
- Todas las posiciones de la RAM funcionan como registros de origen y/o de destino de operaciones matemáticas y otras funciones.
- Una pila de hardware para almacenar instrucciones de regreso de funciones.
- Una relativamente pequeña cantidad de espacio de datos direccionable (típicamente, 256 bytes), extensible a través de manipulación de bancos de memoria.
- El espacio de datos está relacionado con el CPU, puertos, y los registros de los periféricos.
- El contador de programa está también relacionado dentro del espacio de datos, y es posible escribir en él (permitiendo saltos indirectos).

### Microcontroladores AVR

Figura 3: Logotipo de Atmel

Los microcontroladores AVR son una familia de microcontroladores RISC de Atmel inicialmente concebida por dos estudiantes del Norwegian Institute of Technology y desarrollada por Atmel Norway. Los AVR fueron los primeros microcontroladores en incorporar memoria flash, en lugar de la OTP ROM, EPROM o EEPROM habitual en su época.

En este trabajo se utilizará un microcontrolador de Atmel de la familia **AVR**, que son microcontroladores **RISC** con una CPU que sigue la **arquitectura Harvard**, que son aquellos computadores que disponen de una memoria de instrucciones separada de la memoria de datos.

Las características principales de los microcontroladores AVR son las siguientes:

- Están fabricados con tecnología CMOS.
- Las instrucciones se ejecutan en un ciclo de reloj con pipeline.
- Puertos de E/S bidireccionales configurables independientemente pin por pin.
- Timer's. Temporizadores de alta precisión o contadores de pulsos externos
- WatchDog. Monitoriza que el AVR funcione adecuadamente a lo que se esperaba y no se bloquee.
- ISP (In System Programming). Permite realizar la programación del AVR utilizando una interface serial con muy pocos pines.

## 2.2. GNU/Linux

GNU/Linux es un sistema operativo libre *unix-like* creado en 1991, con la unión del kernel Linux lanzado por el finlandés Linus Torvalds y el conjunto de software de GNU.

Diversas empresas, organizaciones y gobiernos ponen a la disposición del usuario las **distribuciones Linux** que son el compendio de un núcleo Linux con una serie de paquetes software incluidos orientados a un determinado sector o a un determinado grupo de usuarios, aunque también existen distribuciones generalistas. Las distribuciones Linux mas conocidas son Debian, Fedora, Ubuntu, openSUSE, Mandriva, Slackware y Gentoo.

Según un informe de IDC, GNU/Linux es utilizado por el 78 % de los principales 500 servidores del mundo, otro informe le da una cuota de mercado de 89 % en los 500 mayores supercomputadores. Con menor cuota de mercado el sistema GNU/Linux también es usado en el segmento de las ordenadores de escritorio, portátiles, ordenadores de bolsillo, teléfonos móviles, sistemas empotrados, videoconsolas y otros dispositivos, en ordenadores de escritorio la cuota de mercado es ligeramente superior al 1 %, pero no se tienen estadísticas confiables y se cree que esta cifra es mayor.

## 2.3. Protocolo USB

El protocolo **USB** o **Universal Serial Bus** es un estándar definido en 1996 por Compaq, NEC, Northern Telecom, IBM, DEC, Intel y Microsoft para proveer de alimentación y comunicación a periféricos externos. Su comunicación es serie y la norma USB es utilizada en más de 2.000 millones de dispositivos cada año.

Una de las características generales del protocolo es que permite **plug and play** y **hot plug**, es decir, permite conectar dispositivos en caliente y que estén listos para funcionar sin tener que reiniciar el PC. Inicialmente (en su versión 1.0) tenia una velocidad de 1.5Mbps. y actualmente en su versión 3.0 se alcanzan velocidades de hasta 5 Gbps.

Existen diversos tipos de conectores USB, que se subdivden en **tipo A** y **tipo B**. En la siguiente figura se muestran los distintos tipos de conectores USB que hay en el mercado:


Figura 4: Tipos de conectores USB

## 2.4. Lenguaje C

**C** es un lenguaje de programación creado en 1972 por **Dennis M. Ritchie** en los Laboratorios Bell como evolución del anterior lenguaje B.

Se trata de un lenguaje fuertemente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

## 2.5. Componentes electrónico básicos

Desde que apareció la electrónica han ido apareciendo componentes electrónicos cada vez más complejos, pero hay una serie de componentes que son básicos en cualquier tipo de dispositivo electrónico.

### Resistencia

Figura 5: Resistencia

Una resistencia es un componente electrónico pasivo que opone cierta dificultad al paso de la corriente eléctrica. Las resistencias tiene una gran variedad de valores expresando en ohmios($\Omega$) y se representa dicho valor con una serie de aros de colores alrededor de la resistencia. Las resistencias pueden ser de varios materiales, como película de carbón, película metálica, cerámicas, etc.

### Condensador

Un condensador es un dispositivo capaz de almacenar una determinada cantidad de electricidad. Se componen de dos superficies conductoras, llamadas armaduras, puestas frente a frente y aisladas entre sí por un material aislante que es llamado dieléctrico. La capacidad de almacenar electricidad es proporcional directamente a la superficie enfrentada; inversamente proporcional a la distancia que separa las armaduras y depende del dieléctrico existente entre ambas. La capacidad de los condensadores se mide en Faradios(F) aunque habitualmente se usan submúltiplos, como el microFaradio($\mu$F), el nanoFaradio(nF) y el picoFaradio(pF).

Existen distintos tipos de condensadores, los más habituales son :

- **Electrolíticos**
  Son polarizados y la primera armadura es un un electrolito. Sus capacidades van desde cientos de nF a varios F.


Figura 6: Condensador electrolítico

- **Cerámicos**
  Son condensadores sin polaridad y usan materiales cerámicos para el dieléctrico. Suelen tener valores desde pF a varios cientos de nF.


Figura 7: Condensador cerámico

- **De película**
  Son condensadores cuyo dieléctrico es una película de poliéster.

Figura 8: Condensador de película

### Circuitos integrados

Un circuito integrado es una pastilla (o chip) muy delgada en la que se encuentran miles o millones de dispositivos electrónicos interconectados, principalmente transistores, aunque también componentes pasivos como resistencias o condensadores. Los circuitos integrados pueden realizar tareas sencillas (como las puertas lógicas) o tareas mucho más complejas (como los microprocesadores).

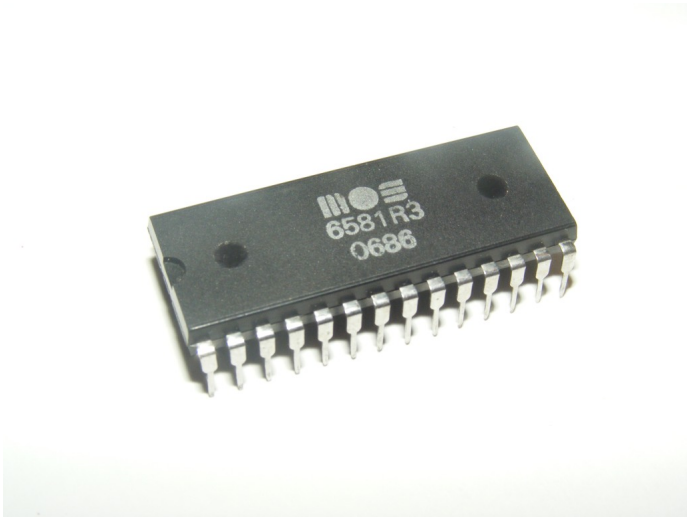


Figura 9: Circuito integrado

## 2.6. Redundancia de Hardware

La redundancia hardware es la **duplicación de elemento físicos** en un sistema para asegurar su funcionamiento a pesar de los **posibles fallos** que puedan surgir durante un uso continuado del mismo. La redundancia hardware se presenta como una solución a los problemas de **protección y confiabilidad**.

Existen varios tipos de redundancia hardware, siendo los principales los siguientes:

- **Dual Modular Redundant (DMR)**
  Se trata de redundancia hardware basada en duplicar elementos para que funcionen en paralelo. Los elementos duplicados tienen que ser del mismo tipo y estar en el mismo estado y para las mismas entradas todos los elementos duplicados deben dar la misma salida esperada. Las salidas se compararán mediante un algoritmo que detectará el fallo de uno de los elementos e intentará ofrecer una salida correcta.

- **Triple Modular Redundancy (TMR)**
  Es un tipo de redundancia hardware para basado en la triplicación del elemento que se desea que sea tolerante a fallos, produciendo una única salida calculada por un algoritmo de votación. Si cualquiera de los tres sistemas fallase, los dos restantes podrían generar la salida y hacer que el fallo fuera transparente, pero en caso del dispositivo que implementa el algoritmo de votación fallase, fallaría todo el sistema. Este tipo de redundancia no solo se aplica al hardware únicamente si no que también es aplicado al software en las programación multiversión (N-version programming).

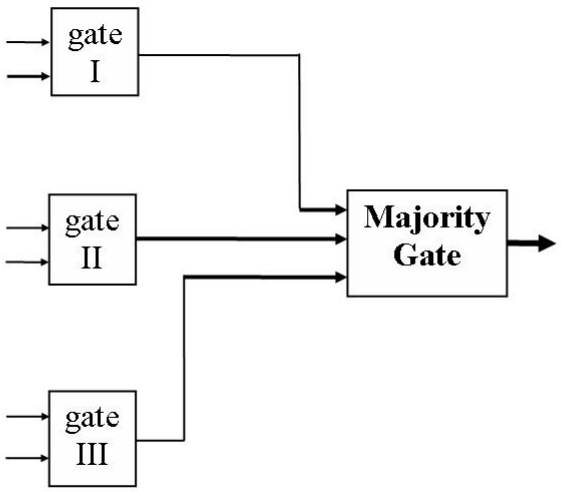


Figura 10: Ejemplo de Triple Modular Redundancy

# 3. Análisis del sistema

En esta sección se define el análisis del sistema a desarrollar, incluyendo la especificación de requisitos de usuario.

En primer lugar se procederá a seleccionar la plataforma de trabajo (tanto hardware como software), para posteriormente reflejar las condiciones del proyecto incluyendo las capacidades generales. Finalmente se especificarán los requisitos de usuario.

Se ha basado en la metodología de desarrollo de la Agencia Espacial Europea (ESA) en su versión Lite[2] con modificaciones, obviando algunas secciones ya que no es el objetivo de este proyecto.

## 3.1. Análisis de la plataforma de trabajo

Debido a que se trata de un proyecto en el que se parte desde cero en cuanto al hardware y en gran parte del software, es necesario barajar varias posibilidades que existen en el mercado para poder realizar el dispositivo.

Se procede a diferenciar el análisis de la plataforma hardware de la plataforma software:

### 3.1.1. Análisis de la plataforma Hardware

El periférico se ha decidido basarlo en un microcontrolador, debido a que ofrece mayor facilidad y comodidad al ingeniero informático frente al diseño basado en lógica discreta.

En el mercado existen bastantes familias de microcontroladores de 8, 16 y 32 bits fabricadas por diversas empresas como Microchip, Atmel, Freescale (antigua Motorola), Intel, NXP (antiga Philips), Renesas, ST, Zilog, Texas Instruments, etc.

Actualmente las dos familias de microcontroladores más extendidas y con más soporte de la comunidad son los **PIC de Microchip** y los **AVR de Atmel**.

En el capitulo de estado del arte se han descrito las familias de microcontroladores de 8 bits PIC y AVR, y a continuación se muestra una tabla comparando dos micrcontroladores concretos, de características similares, ambos válidos para usarlos como núcleo del periférico. Son el **ATMEGA168** de Atmel y el **PIC18F2410** de Microchip.

| | **ATMEGA168** | **PIC18F2410** |
|---|---|---|
| Pines | 28 | 28 |
| Flash | 16Kb. | 16Kb |
| RAM | 1024bytes | 768bytes |
| EEPROM | 512bytes | 0bytes |
| MIPS | 12MIPS@12Mhz. | Hasta 10MIPS |
| E/S | 23 pines. | 25 pines |
| ADC | 8 canales | 10 canales |
| Voltaje | 1.8-5.5V | 2.0-5.5V |
| Precio | $\approx$ 2.5€ | $\approx$3.3€ |

Tabla 1: Comparativa microcontroladores posibles

Los criterios para encontrar estas dos alternativas son los siguientes:

- **Número de pines**
  Era necesario al menos un microcontrolador con al menos 15 pines de E/S para poder controlar la pantalla, los pulsadores, los sensores de temperatura y el puerto USB.

- **Flash**
  A priori, no se sabe la cantidad necesaria de memoria flash, pero 16Kb. es una cantidad suficiente para aplicaciones de este tipo.

- **ADC**
  Debido a que se van a tener tres sensores de temperatura, será necesario que el microcontrolador tenga al menos 3 canales de ADC (Conversor analógico-digital).

- **Voltaje**
  El microcontrolador se debería poder alimentar a 5V ya que es la tensión que proporciona el puerto USB.

Aparte de los criterios relacionados con las características del microcontrolador, para poder seccionarlo es muy importante examinar los lenguajes de programación disponibles así como los entornos de desarrollo.

En ambos casos, para los micontroladores PIC y AVR, hay compiladores de los dos lenguajes de programación más utilizados en el mundo de los microntroladores (aparte de ensamblador), **BASIC** y **C**.

En el caso de los microcontroladores PIC, existe el compilador de BASIC más usado es PICBASIC PRO, cuya versión para estudiantes cuesta 50$. En cuanto a compiladores de C, existen el CSS cuya licencia cuesta 50$ incluyendo sólo el compilador de línea de comandos y 350$ incluyendo el entorno de desarrollo. Por otro lado, la propia Microchip ofrece un entorno de desarrollo con compilador de C llamado MPLAB de manera gratuita en su versión básica pero con niveles de optimización de código limitados. Los tres compiladores están disponibles únicamente para Microsft Windows.

Por su lado, los microcontroladores AVR disponen también de un opción comercial para desarrollar en BASIC llamara BASCOM-AVR desde 89€para Windows. Con respecto a C, los AVR disponen del compilador GCC gracias a las librerías AVR Libc, siendo este libre, gratuito,

multiplataforma y con buen nivel de optimización de código. Además de que existen multitud de librerias extra como la V-USB que permite una comunicación sencilla mediante USB.

Después de exponer las dos posibilidades barajadas como potenciales microcontroladores para la implementación del periférico se ha decidido usar el microcotrolador de la familia AVR de Atmel, **ATMEGA168**.

Ha sido seleccionado por los siguientes motivos:

- **Su excelente y gratuito compilador de C**
  Este ha sido uno principales motivos para su selección, ya que la disponibilidad de GCC para AVR permitirá poder compilar el código en distintas plataformas y la disponibilidad de la librería V-USB para poder realizar la comunicación USB, aparte de la gran potencia y versatilidad que ofrece.
- **Buen rendimiento a bajo coste**
  Aunque el precio sea ligeramente inferior al su rival de Microchip, la potencia que da es superior (1 MIPS por cada Mhz.)
- **Familiaridad con la arquitectura**
  Debido a la familiaridad previa con la arquitectura AVR adquirida por el uso de Arduino en la asignatura de Sistemas Informáticos.

### 3.1.2. Análisis de la plataforma Software

Después de seleccionar la plataforma hardware del periférico, se procederá a elegir la plataforma software. La selección del la plataforma software la podemos dividir en dos subselecciones: **Selección del Sistema Operativo** y **Selección del lenguaje de programación** (para el firmware y para el programa cliente del ordenador).

#### Análisis del Sistema Operativo

Dado que este periférico está principalmente orientado a sistemas empotrados y servidores, el sistema operativo elegido será el mas utilizado en estos equipos.

Como puede verse en la figura 11 según W3Techs [1] casi dos tercios de la cuota de mercado en servidores corresponde a sistemas operativos Unix y Unix-like (como GNU/Linux).

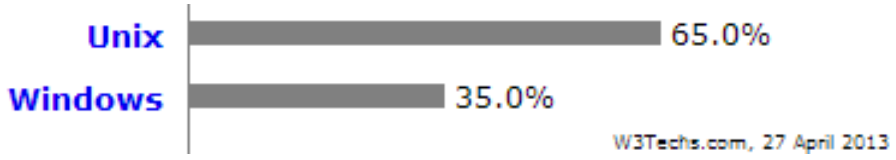


Figura 11: Cuota de mercado SO en servidores (a fecha de 27/04/2013)

La cuota de mercado de Unix desglosada puede verse en la figura 12 donde Linux va en cabeza seguido de desconocido (que pueden tratarse también de sistemas Linux u otros Unix, que no han podido ser detectados).

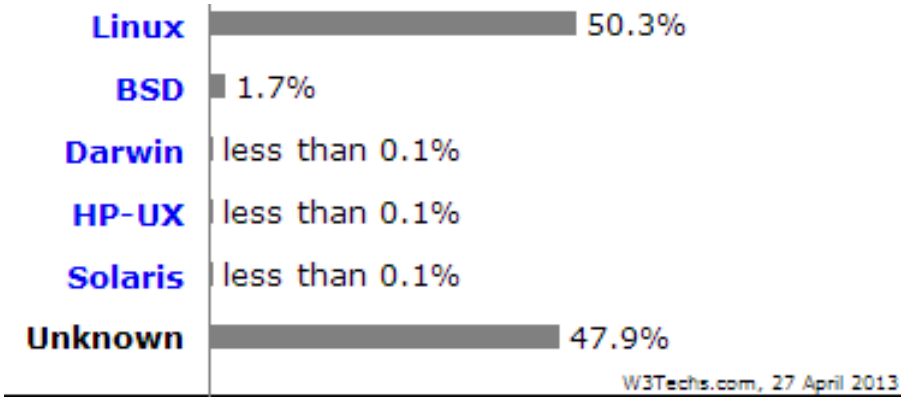


Figura 12: Cuota SO Unix en servidores (a fecha de 27/04/2013)

A la vista de estos datos, el sistema operativo será por tanto un **Unix-like** y específicamente **GNU/Linux**.

**Análisis del lenguaje de programación**

La selección del lenguaje de programación se puede dividir en dos decisiones a tomar, la **selección del lenguaje del firmware del microcontrolador** y la **selección del lenguaje del programa cliente** (en nuestro caso de un demonio).

El lenguaje para el **fimware del microcontrolador**, será **C** debido a que uno de los motivos por los que se ha elegido el microcontrolador AVR es por el buen compilador de C que dispone (GCC) y las librerías que hay disponibles.

En cuanto al lenguaje de programación del **demonio**, dado a que se ha seleccionado como sistema operativo Linux, el lenguaje escogido será también **C**, dado que la implementación de demonios se realiza habitualmente en C en Linux.

## 3.2. Descripción General

En esta sección tal y como se describe en el estándar de la esa, es importante señalar las capacidades y restricciones generales del sistema a diseñar que servirán como punto de partida de los requisitos de usuario.

### 3.2.1. Capacidades generales

Las capacidades generales del sistema serían las siguientes:

- El sistema se trata de un periférico de monitorización y control para sistemas empotrados, servidores y cualquier ordenador en general.

- El sistema será capaz mostrar una serie de **menús** en una **pantalla LCD**, cuyos menús serán manejados por **3 pulsadores**(subir, bajar y aceptar).

- Se podrá leer el valor de la temperatura ambiente mediante sus **sensores de temperatura** y ser mostrado en pantalla.

- El sistema será capaz de **comunicarse bidireccionalmente** con un ordenador o sistema empotrado vía **USB**, permitiendo mostrar datos del ordenador en la pantalla LCD y datos del periférico en el ordenador.

- El sistema se podrá manejar de **forma intuitiva** (con los menús y pulsadores) sin necesidad de conocimiento previo.

### 3.2.2. Restricciones generales

En cuanto a las restricciones generales del sistema a desarrollar, serían las siguientes:

- La comunicación dispositivo-ordenador se hará mediante un puerto USB que cumpla con la norma **USB 1.1** (o superior).

- El **microcontrolador** será de **Atmel** y con una arquitectura de 8 bits.

- La pantalla del dispositivo será una pantalla **LCD basada en caracteres**. Los menús deberán estar cargados en el propio microcontrolador.
- El firmware y el demonio estarán escritos en el **lenguaje C**.

- El consumo del periférico debe ser **inferior a 500 mA**. para que pueda alimentarse sin problemas por el puerto USB del ordenador.

## 3.3. Requisitos de Usuario

En esta sección se especificarán los requisitos de usuario con el objetivo de definir, concretar, ordenar y catalogar las necesidades que se tienen para este proyecto.

La plantilla de los requisitos es la siguiente:

| **Identificador** | **XXX-YY** |
|---|---|
| **Título** | |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | |
| **Fuente** | |
| **Descripción** | |
| | |

Tabla 2: Plantilla de requisitos

Siendo cada campo lo siguiente:

- **Identificador:** código único e identificativo del requisito. Compuesto por dos partes, XXX donde se indica el código del tipo de requisito (por ejemplo, RUC es para los Requisitos de Usuario de Capacidad) y YY que indica el número de requisito.
- **Título:** nombre único del requisito.
- **Prioridad:** indica la importancia del requisito. Los valores posibles son Alta, Media y Baja.
- **Necesidad:** indica la importancia del requisito desde el punto de vista del cliente. Los valores que puede tomar este campo son esencial, conveniente u opcional.
- **Fuente:** es la procedencia del requisito.
- **Descripción:** aclaración del requisito.

### 3.3.1. Requisitos de Capacidad

| **Identificador** | **RUC-01** |
|---|---|
| **Título** | Interfaz de usuario intuitiva |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| La interfaz de usuario del periférico debe ser intuitiva para que pueda ser usada por un usuario sin experiencia en el mismo. | |

Tabla 3: RUC-01

| **Identificador** | **RUC-02** |
|---|---|
| **Título** | Enviar la temperatura del dispositivo al PC |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El dispositivo será capaz de enviar la temperatura de los sensores al PC. | |

Tabla 4: RUC-02

| **Identificador** | **RUC-03** |
|---|---|
| **Título** | Enviar datos del PC al dispositivo |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El sistema será capaz de enviar datos del PC al dispositivo. | |

Tabla 5: RUC-03

| **Identificador** | **RUC-04** |
|---|---|
| **Título** | Mostrar la temperatura en la pantalla del dispositivo |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El sistema será capaz de mostrar en la pantalla del dispositivo la temperatura leída de los sensores. | |

Tabla 6: RUC-04

| **Identificador** | **RUC-05** |
|---|---|
| **Título** | Mostrar los datos recibidos en la pantalla del dispositivo |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El sistema será capaz de mostrar en la pantalla del dispositivo los datos recibidos del PC. | |

Tabla 7: RUC-05

| **Identificador** | **RUC-06** |
|---|---|
| **Título** | Alimentación desde el PC |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El periférico se alimentará desde el PC. | |

Tabla 8: RUC-06

| **Identificador** | **RUC-07** |
|---|---|
| **Título** | Recuperación entre conexión y desconexión |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El ordenador debe ser capaz de recuperarse entre conexiones y desconexiones del periférico. | |

Tabla 9: RUC-07

| **Identificador** | **RUC-08** |
|---|---|
| **Título** | Respuestas a las peticiones del dispositivo |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El ordenador debe ser capaz de responder a las peticiones realizadas por el dispositivo. | |

Tabla 10: RUC-08

| **Identificador** | **RUC-09** |
|---|---|
| **Título** | Historifiación de la temperatura |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El ordenador debe ser capaz de historificar la temperatura recibida del periférico. | |

Tabla 11: RUC-09

### 3.3.2. Requisitos de Restricción

| **Identificador** | **RUR-01** |
|---|---|
| **Título** | Lenguaje de programación |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El lenguaje de programación usado será C. | |

Tabla 12: RUR-01

| **Identificador** | **RUR-02** |
|---|---|
| **Título** | Sistema Operativo |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El sistema operativo sobre el que se ejecutará el demonio será GNU/Linux. | |

Tabla 13: RUR-02

| **Identificador** | **RUR-03** |
|---|---|
| **Título** | Conexión USB |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| La conexión entre el dispositivo y el PC debe ser mediante USB. | |

Tabla 14: RUR-03

| **Identificador** | **RUR-04** |
|---|---|
| **Título** | Idioma de la aplicación |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El idioma de toda la parte de la interfaz de usuario será el español. | |

Tabla 15: RUR-04

| **Identificador** | **RUR-05** |
|---|---|
| **Título** | Interacción con el usuario |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| La interacción del periférico con el usuario será mediante menús controlados por pulsadores. | |

Tabla 16: RUR-05

| **Identificador** | **RUR-06** |
|---|---|
| **Título** | Consumo acotado |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El periférico deberá consumir menos de 500mA. | |

Tabla 17: RUR-06

### 3.3.3. Requisitos Hardware

| **Identificador** | **RHW-01** |
|---|---|
| **Título** | Microcontrolador |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El microcontrolador utilizado será un Atmega168 DIL de Atmel. | |

Tabla 18: RHW-01

| **Identificador** | **RHW-02** |
|---|---|
| **Título** | Pantalla LCD |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| La pantalla LCD utilizada será una pantalla de 20x4 caracteres, monocroma y retroiluminada. | |

Tabla 19: RHW-02

| **Identificador** | **RHW-03** |
|---|---|
| **Título** | Sensores de temperatura |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| Los sensores de temperatura serán unos LM35 de National Semiconductor. | |

Tabla 20: RHW-03

| **Identificador** | **RHW-04** |
|---|---|
| **Título** | Conexión con el ordenador |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| El periférico tendrá un conector USB hembra tipo B. | |

Tabla 21: RHW-04

| **Identificador** | **RHW-05** |
|---|---|
| **Título** | Pulsadores |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | Cliente |
| **Descripción** | |
| Se utilizará pulsadores momentáneos normalmente abiertos. | |

Tabla 22: RHW-05

# 4. Diseño del sistema

En este capítulo se procederá a detalla el diseño del sistema, tanto de la parte hardware como de la parte software.

## 4.1. Diseño Hardware

En este subapartado se expondrá el diseño hardware, viendo en primer lugar el diseño del circuito electrónico del periférico, para posteriormente especificar el diseño de la placa de circuito impreso (PCB).

### 4.1.1. Diseño del circuito

El primer paso de desarrollo de un dispositivo hardware es el diseño del circuito electrónico. Por lo que en está sección se mostrará y explicará el circuito realizado desde alto nivel.

El circuito del periférico a alto nivel se puede ver en el diagrama de bloques de la figura 13.

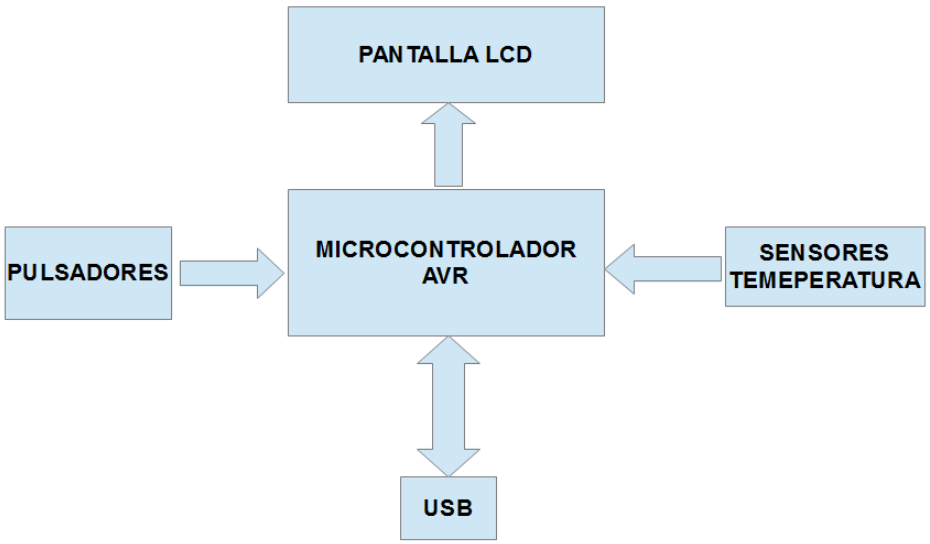


Figura 13: Diagrama de bloques del circuito

Como puede observarse el núcleo del sistema es el **microcontrolador AVR**, que controlara toda la comunicación bidireccional por **USB** además de leer las pulsaciones de los **pulsadores**. Por otro lado, dispone de varios **sensores de temperatura**, que el microcontrolador se encargará de leer cuando sea necesario y de mostrar la temperatura (además de los menús y otros datos recibidos por USB) en la **pantalla LCD**.

A continuación, se procederá a detallar el diseño del circuito según los bloques funcionales que se han expuesto.

**Microcontrolador**

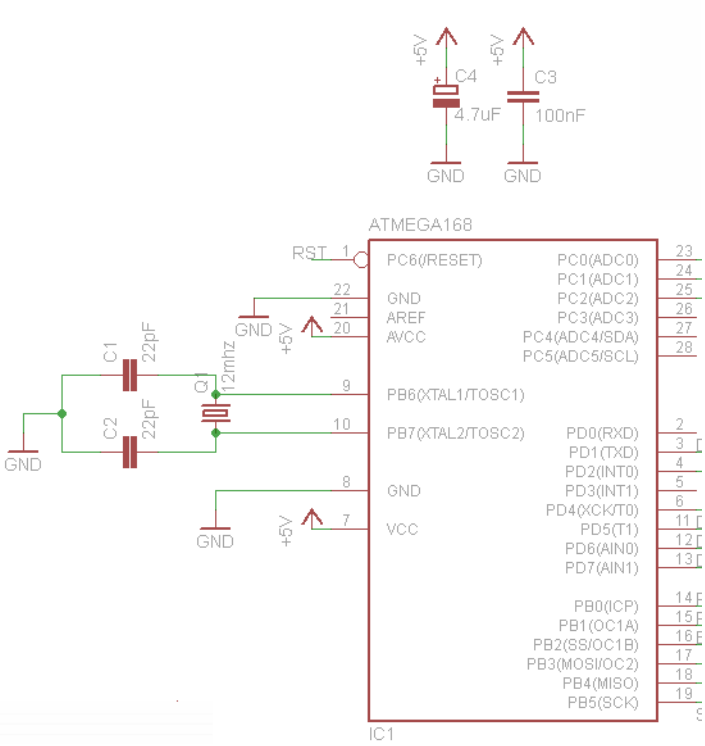


Figura 14: Microcontrolador AVR

En la figura 14 se puede observar el microcontrolador, el corazón del periférico, conectado a las señales de alimentación. También se puede apreciar el oscilador formado por un cristal de cuarzo de **12 Mhz**. y dos condensadores de 22pF. Se ha elegido esa frecuencia para el cristal, ya que es la frecuencia mínima necesaria para el funcionamiento de la librería V-USB para realizar la comunicación USB. El requisito hardware asociado es el *RHW-01*.

**Pantalla LCD**

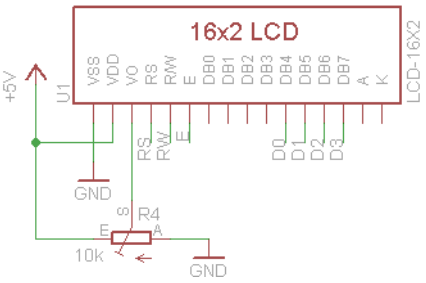


Figura 15: Pantalla LCD

En la figura 15 se puede ver la pantalla de cristal líquido (LCD) de 4 líneas de 20 caracteres por línea monocroma con retroilumnación. La pantalla está gobernada por un controlador compatible con el HD44780 que soporta dos modos de funcionamiento, de **4 bits** y de **8 bits**. Se ha

escogido el modo de funcionamiento de **4 bits** debido a que usa menos pines de entrada/salida. En el esquema aparece indicada como pantalla de 16x2 en lugar de 20x4 ya que el pineado de la pantalla es el mismo y el programa de diseño no disponía la "pieza"para la pantalla de 20x4. El requisito hardware asociado es el *RHW-02*.

**Sensores de temperatura**

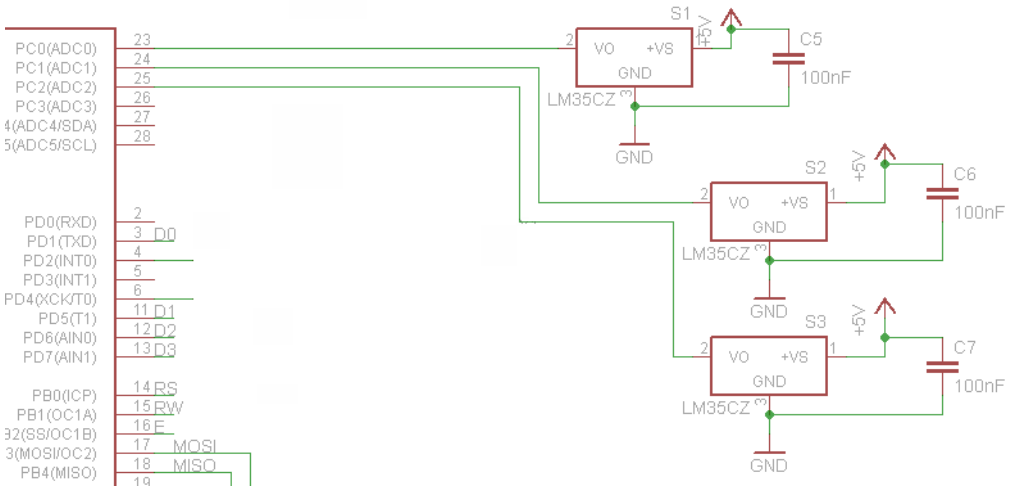


Figura 16: Sensores de temperatura

El sensor de temperatura elegido ha sido el **LM35** de National Semiconductor, que tiene un rango de temperatura de **-55º a +150ºC** con una precision de 1ºC en el peor caso. Se ha escogido este sensor por su **salida lineal** (cada grado equivale a 10mV) y porque no requiere ningún tipo de calibrado.

Cada uno de los tres sensores se ha conectado a una entrada analógica del microcontrolador y se ha alimentado el sensor con 5V. A su vez, se añadido un condensador de desacople de 100nF para eliminar el posible ruido. El requisito hardware asociado es el *RHW-03*.

**Pulsadores**

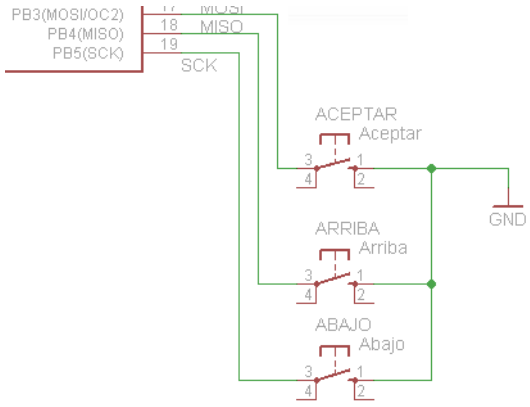


Figura 17: Pulsadores de control

Para el manejo de los menús en pantalla se ha optado por poner **3 pulsadores normalmente abiertos**, para manejar las funciones de **arriba**(para subir a la opción anterior), **abajo** (para bajar a la siguiente opción) y **aceptar** (para seleccionar la opción actual).

No se han colocado resistencias de *pullup* ya que se usarán las internas que tiene el microcontrolador en cada pin de entrada/salida. El requisito hardware asociado a los pulsadores es el *RHW-05*.

**USB**

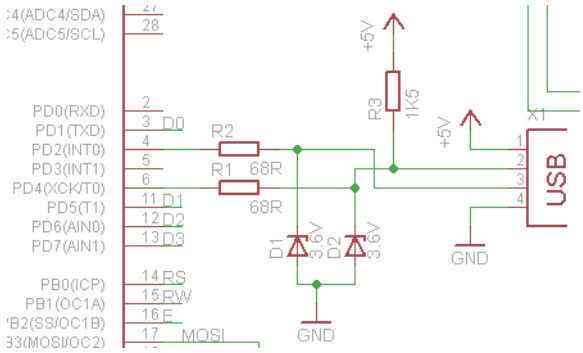


Figura 18: USB

Para conectar el microcontrolador por USB es necesario usar una serie de componentes adicionales para adecuar las señales a la del puerto USB. Para adecuar las señales de datos al voltaje adecuado (3.6V) se usan dos **diodos zener de 3.6V**, D1 y D2, y dos resistencias de **68ohm**, R1 y R2. Además en la linea de datos D- se ha colocado una resistencia de **pullup de 1k5**, R3; que sirve para el ordenador reconozca correctamente el dispositivo.

También cabe destacar que la alimentación del circuito es obtenida del puerto USB .El requisito hardware asociado al puerto USB es el *RHW-04*.

En la figura 19 puede verse el circuito completo con todas las partes que se acaban de explicar. Se puede apreciar en la parte izquierda del circuito el conector AVR-SPI que sirve para poder programar el microcontrolador.

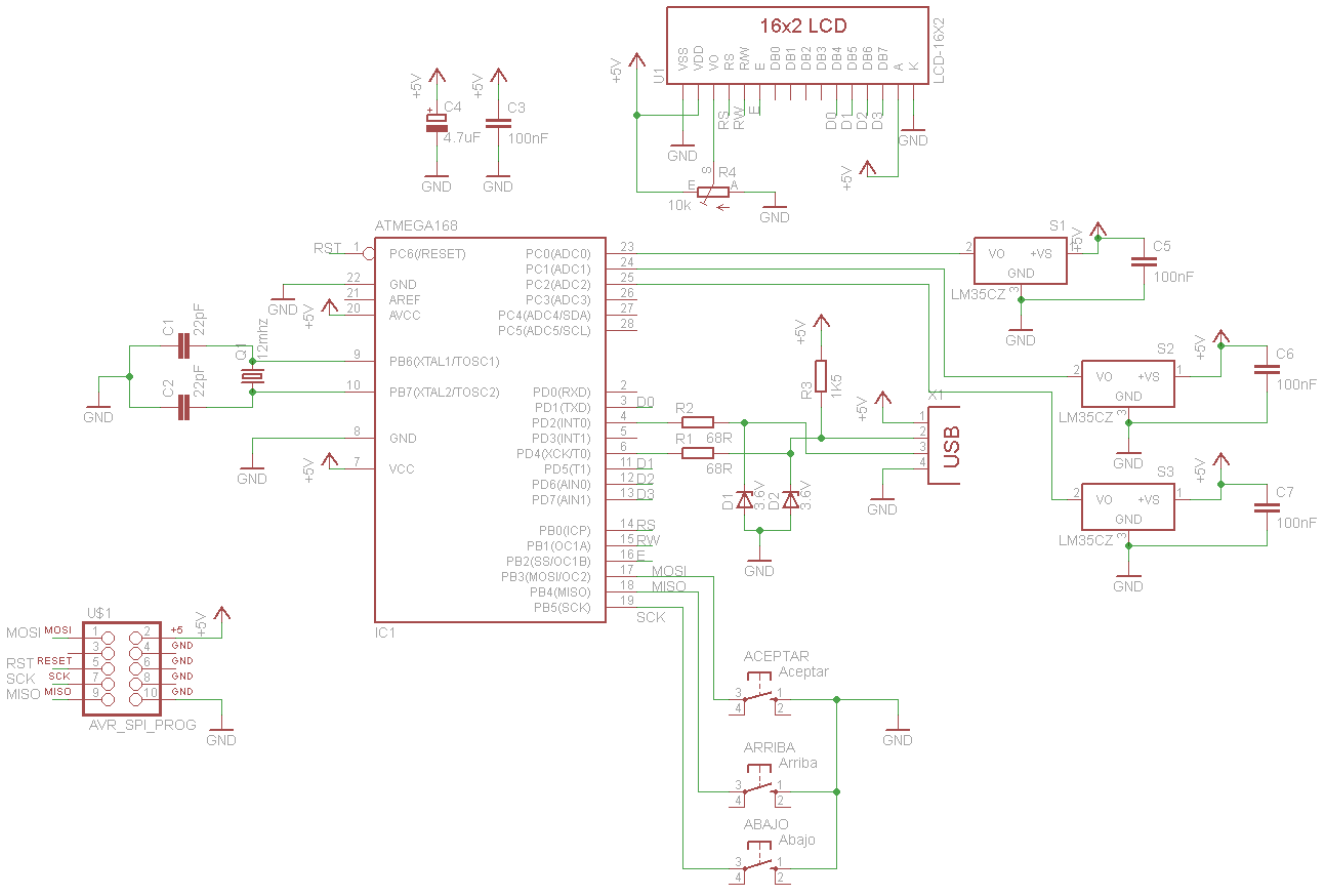


Figura 19: Circuito completo

### 4.1.2. Diseño de la PCB

En esta sección se va a proceder a realizar el diseño de la **placa de circuito impreso o PCB**. Una vez realizado el diseño del circuito y comprobado su funcionamiento mediante el montaje de un prototipo (como se podrá observar en la sección 5.1), se procede a diseñar la placa de circuito impreso.

Para realizar el diseño se utilizará un programa de diseño electrónico automatizado (EDA), que en este caso será el **Cadsoft Eagle**.

Pero antes de comenzar el diseño con Eagle, habría que definir el tamaño de la placa. Para ello, como puede verse en la figura 20, se ha realizado un boceto en papel, planteando los principales componentes del periférico.

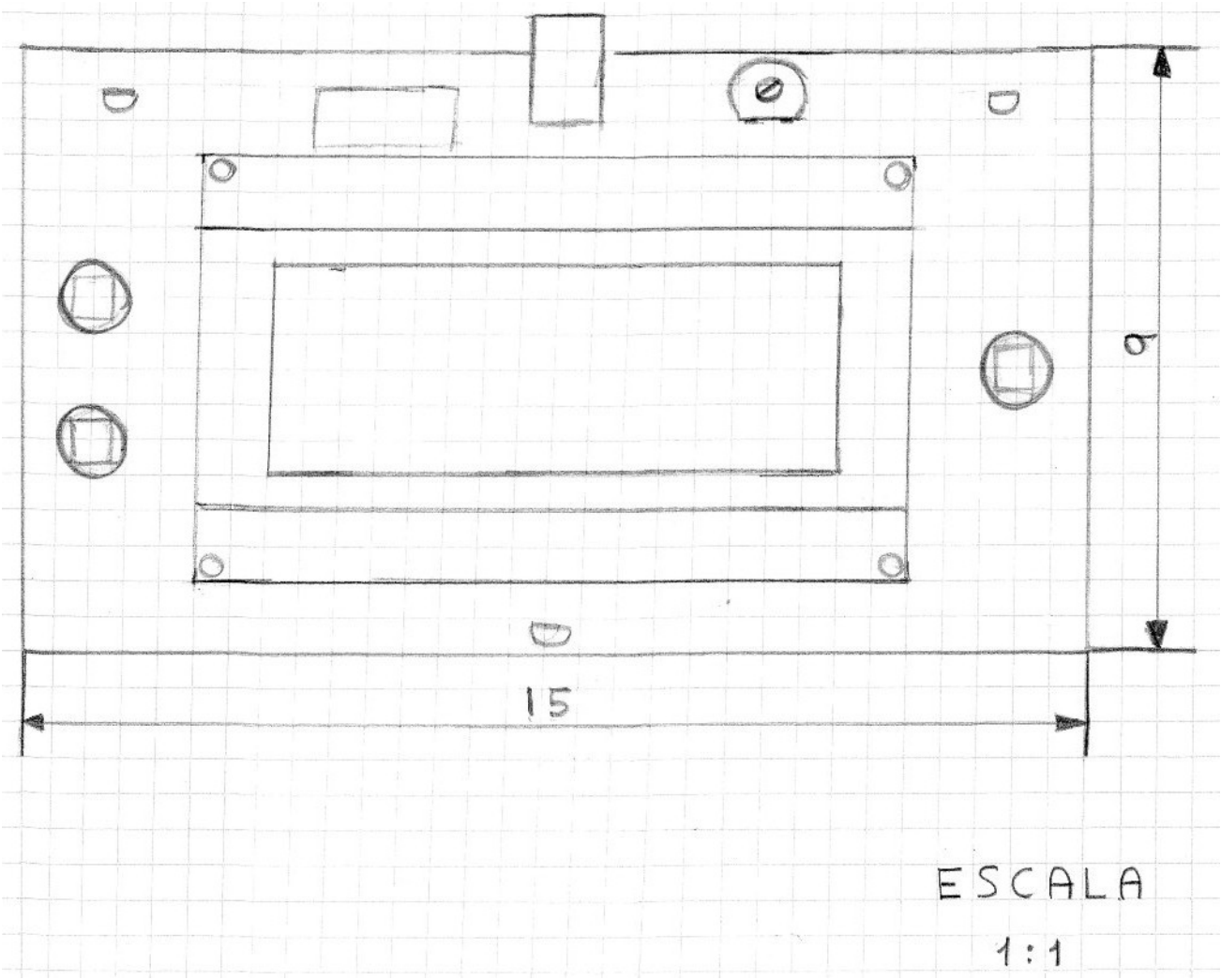


Figura 20: Boceto de la PCB (medidas en cm.)

Para aprovechar el espacio de la placa y reducir el tamaño de la placa, el micrcontrolador, y algunos componentes asociados como el cristal de cuarzo, se situará debajo de la pantalla LCD.

Gracias al boceto se ha podido obtener el **tamaño de la placa** necesario, que en este caso es de **15x9cm**. Es un factor importante a tener en cuenta ya que repercutirá en el coste de fabricación de la placa, dado que el fabricante va por tramos de tamaño. En este caso se entra en el tramo de 15x10cm.

El diseño del circuito también fue realizado en Eagle, por lo que pasar al diseño de la PCB será mas sencillo ya que se mantendrán todas las conexiones entre los componentes, por lo que únicamente habría que **definir el tamaño** de la placa, **colocarlos** en en lugar de la placa adecuado y **enrutar** las pistas de la PCB, aparte de añadir los planos de masa y **serigrafía**.

Para la colocación de los componentes se ha seguido el boceto inicial, colocando el micro-controlador, el cristal de cuarzo, las resistencias, diodos y los condensadores de desacople y del cristal bajo la pantalla LCD, para aprovechar el espacio. En la parte superior se ha conectado el conector USB hembra para conectar el cable que lo comunicará con el PC.

Por otro lado, los pulsadores se han colocado **tipo mando de videoconsola**, los de dirección en el lado izquierdo y el de aceptar al lado derecho, para que el periférico pueda cogerse con dos manos.

Los **sensores de temperatura** se han colocado de la siguiente manera: los dos primeros en las dos esquinas superiores y el tercero centrado en la parte inferior de manera que el calor de las manos al sujetar el periférico no pueda influir en las medidas tomadas por los sensores. Al lado de cada sensor se ha colocado el condensador de desacople correspondiente.

Finalmente se ha colocado el condensador de filtro arriba de la pantalla así como el potenciómetro de ajuste del contraste de la misma y el conector AVR-SPI para programar el microcontrolador.

A continuación se ha procedido a rutear las pistas en la placa, se ha decidido utilizar dos caras, ya que como la placa se va a mandar a un fabricante y no se va a realizar de forma casera, no supone un handicap y por otro lado, el coste de fabricación es lo mismo para una que para dos caras en la empresa elegida, por lo que no incrementará los costes. Eagle permite enrutar la placa de manera manual o de manera automática, por lo que se ha procedido a enrutar la placa de manera automática con algún retoque posterior del ruteado de manera manual.

Posteriormente se han añadido dos planos en la capa superior e inferior, siendo el de la capa superior un **plano de 5V** y el de la capa inferior un **plano de masa (GND)**. Se han puesto estos planos porque es recomendable para evitar las posibles corrientes parásitas.

En último lugar, se ha colocado la serigrafía del copyright, versión, etc. así como colocar varios nombres en otro lugar para evitar su solapamiento. La serigrafía de los componentes la genera automáticamente Eagle a partir de lo especificado en el esquema.

En la figura 21 puede verse el diseño de la placa terminado.

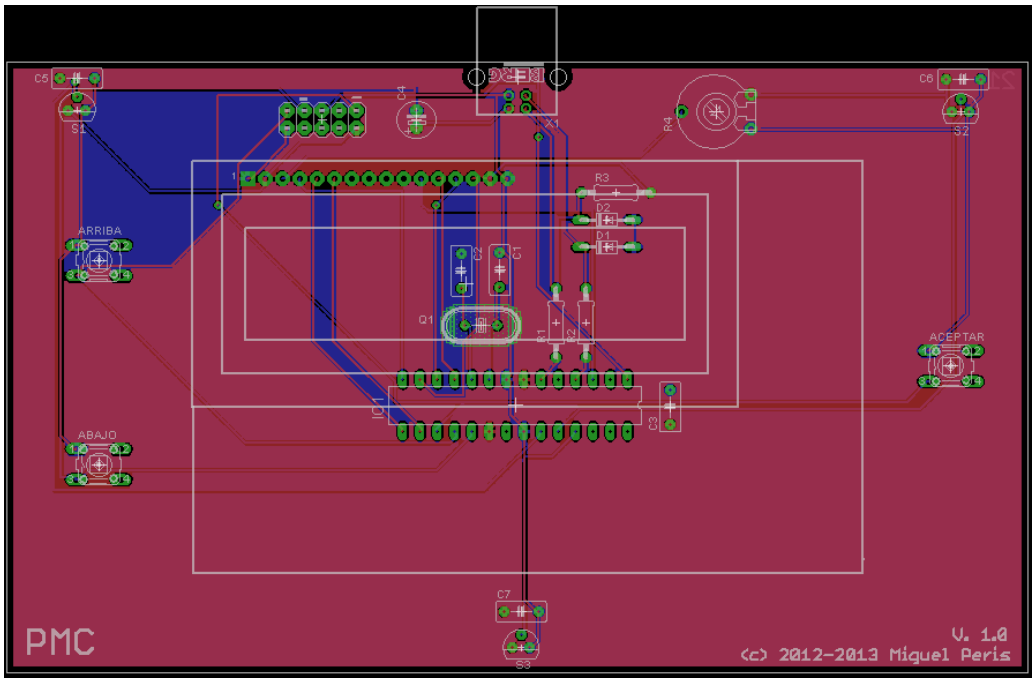


Figura 21: Diseño de la PCB

**Seeedstudio**[4] provee un servicio profesional de fabricación de PCB económico, desde 9.99$ y para tiradas pequeñas, desde 5 placas, por lo que se ha decido pedir las placas a este fabricante en lugar de fabricarlas de manera casera.

En las figuras 22 y 23 pueden verse las placas ya fabricadas, por ambas caras.

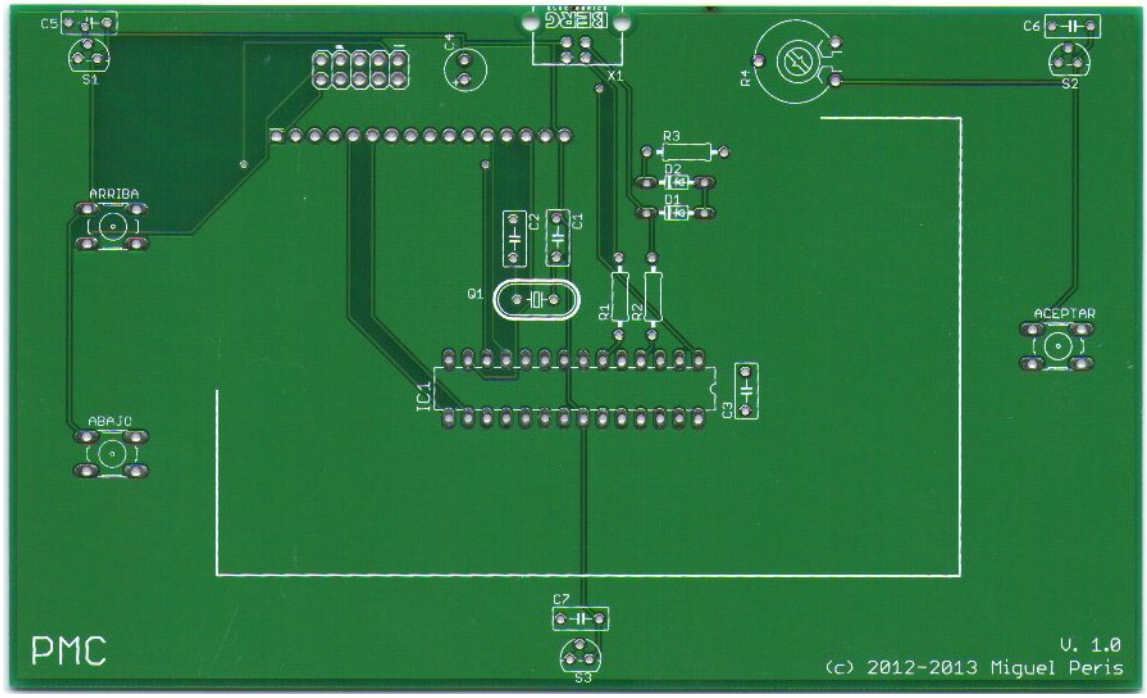


Figura 22: PCB fabricada (cara superior)

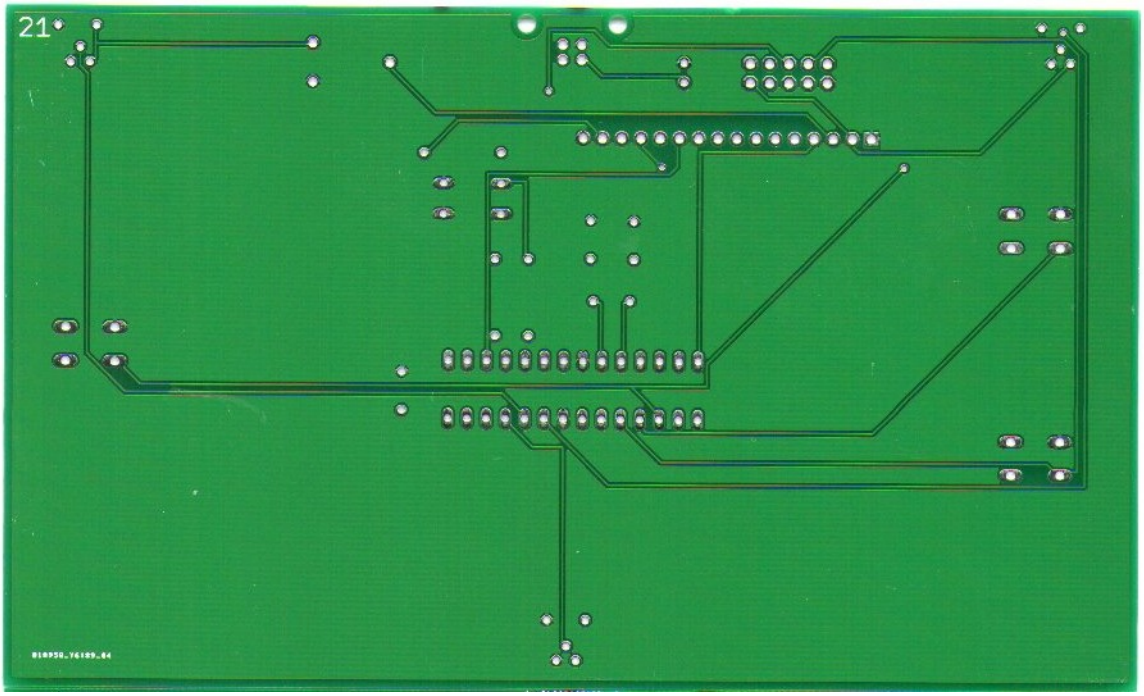


Figura 23: PCB fabricada (cara inferior)

## 4.2. Diseño Software

En esta sección se desarrollará el diseño de la parte software del proyecto, tanto el firmware del microcontrolador como del demonio Linux. Además se definirán los requisitos software.

### 4.2.1. Requisitos Software

A continuación se definen los requisitos software del proyecto, siguiendo la misma plantilla especificada en los requisitos de usuario.

**Requisitos Funcionales**
Los requisitos software funcionales son los siguientes:

| **Identificador** | **RSF-01** |
|---|---|
| **Título** | Uso de menús |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-01 |
| **Descripción** | |
| Se usaran una serie de menús para manejar el periférico. | |

Tabla 23: RSF-01

| **Identificador** | **RSF-02** |
|---|---|
| **Título** | Comunicación USB |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-02 y RUC-03 |
| **Descripción** | |
| Se realizará la comunicación bidireccional entre el dispositivo y ordenador por protocolo USB. | |

Tabla 24: RSF-02

| **Identificador** | **RSF-03** |
|---|---|
| **Título** | Muesta de datos en la pantalla LCD |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-04 y RUC-05 |
| **Descripción** | |
| Se mostrará la temperatura leída de los sensores y los datos recibidos del ordenador en la pantalla LCD del dispositivo. | |

Tabla 25: RSF-03

| **Identificador** | **RSF-04** |
|---|---|
| **Título** | Alimentación del periférico |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-06 |
| **Descripción** | |
| El periférico se alimentará por el puerto USB. | |

Tabla 26: RSF-04

| **Identificador** | **RSF-05** |
|---|---|
| **Título** | Demonio en el ordenador |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-08 |
| **Descripción** | |
| Exisitirá un demonio en el ordenador que responderá las peticiones del periférico. | |

Tabla 27: RSF-05

| **Identificador** | **RSF-06** |
|---|---|
| **Título** | Capacidad de recuperación entre desconexiones |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-07 |
| **Descripción** | |
| El demonio será capaz de recuperarse tras una conexión-desconexión del periférico. | |

Tabla 28: RSF-06

| **Identificador** | **RSF-07** |
|---|---|
| **Título** | Historificación de la temperatura |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-09 |
| **Descripción** | |
| El demonio leerá la temperatura del dispositivo y la historificará en el ordenador. | |

Tabla 29: RSF-07

| **Identificador** | **RSF-08** |
|---|---|
| **Título** | Comandos de control entre dispositivo-ordenador |
| **Prioridad** | ☒Alta ☐ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-09 |
| **Descripción** | |
| El ordenador y el periférico se comunicarán para realizar peticiones de información mediante una serie de comandos basados en un caracter. | |

Tabla 30: RSF-08

**Requisitos no Funcionales**

Los requisitos software no funcionales se subdividen en Requisitos de Interfaz, Requisitos de Rendimiento, Requisitos Operacionales y Requisitos de Portabilidad.

**Requisitos de Interfaz**

Los Requisitos de Interfaz son los siguientes:

| **Identificador** | **RSI-01** |
|---|---|
| **Título** | Parte diferenciada para menús y resultados |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-01 |
| **Descripción** | |
| En la pantalla LCD estará diferenciada la zona de menús de la zona de resultados. | |

Tabla 31: RSI-01

| **Identificador** | **RSI-02** |
|---|---|
| **Título** | Regreso a menús anteriores |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-01 |
| **Descripción** | |
| En los submenús existirá la opción de volver al menú superior. | |

Tabla 32: RSI-02

| **Identificador** | **RSI-03** |
|---|---|
| **Título** | Los menús se manejaran con pulsadores |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | RUC-01 |
| **Descripción** | |
| Los menús se podrán manejar con tres pulsadores, uno para subir al menú anterior, otro para bajar al menú siguiente y uno para aceptar el menú actual. | |

Tabla 33: RSI-03

**Requisitos de Rendimiento**

Los Requisitos de Rendimiento son los siguientes:

| **Identificador** | **RSR-01** |
|---|---|
| **Título** | Tiempo de respuesta aceptable |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | Conveniente |
| **Fuente** | - |
| **Descripción** | |
| La respuesta de una petición del dispositivo debe llegar en un tiempo inferior a 3 segundos. | |

Tabla 34: RSR-01

**Requisitos Operacionales**

Los Requisitos Operacionales son los siguientes:

| **Identificador** | **RSO-01** |
|---|---|
| **Título** | Entorno operacional |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | - |
| **Descripción** | |
| El demonio correrá sobre el sistema operativo GNU/Linux. | |

Tabla 35: RSO-01

| **Identificador** | **RSO-02** |
|---|---|
| **Título** | Demonio iniciado en el arranque |
| **Prioridad** | ☐Alta ☒ Media ☐Baja |
| **Necesidad** | Esencial |
| **Fuente** | - |
| **Descripción** | |
| El demonio se iniciará al arrancar el ordenador. | |

Tabla 36: RSO-02

### 4.2.2. Diseño del firmware

El microcontrolador para funcionar necesita tener un firmware en su memoria de instrucciones, en el caso del Atmega168, memoria flash. El firmware se va a encargar de las siguientes tareas:

- Leer las pulsaciones de las teclas de control.
- Gestionar la pantalla LCD y los menús que se mostrarán en ella.
- Controlar la comunicación bidireccional por USB.
- Leer los valores de los sensores de temperatura.
- Mostrar la temperatura según los distintos algoritmos de votación.

Cuando el micrcontrolador recibe alimentación por USB empieza ejecutar el código del firmware de manera infinita. En la figura 24 se muestra el diagrama de flujo de la ejecución principal del código.

Una vez comienza a ejecutar el código el microcontrolador se realiza una **configuración inicial** de la pantalla LCD, puerto USB, ADC y puertos de entrada, para posteriormente entrar en el bucle principal. Una vez dentro del bucle principal se muestra por pantalla la posición actual del menú y se comprueba si se ha pulsado algún botón. Si es así realiza acción pertinente, si se ha pulsado arriba cambia al menú superior(que se mostrará en la próxima iteración del bucle), si se pulsa abajo cambiará al menú inferior y si se pulsa aceptar realizará la acción correspondiente al menú que se muestra actualmente por pantalla.

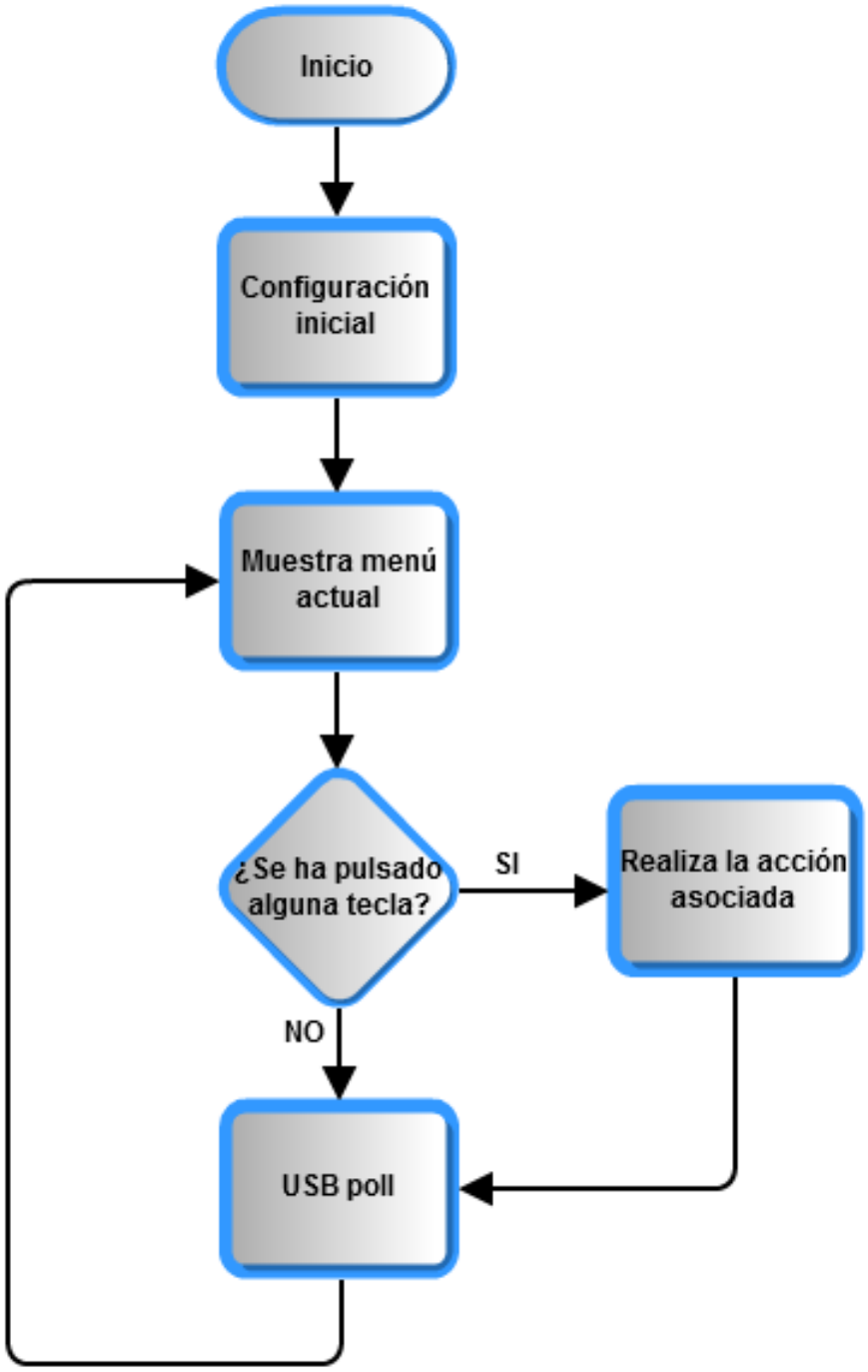


Figura 24: Diagrama de flujo

### Algoritmos de votación

Para aumentar la precisión de la lectura de temperatura y que sea **tolerante a fallos**, se ha optado por la **redundancia hardware** de los sensores de temperatura, en concreto se utilizará una redundancia hardware triple modular(**Triple Modular Redundancy**). Por consiguiente, se ha decidido poner en el periférico **tres sensores de temperatura**.

En el firmware del dispositivo se desarrollarán varios **algoritmos de votación** que toman los valores de los tres sensores y muestran un único valor. Los algoritmos a desarrollar son los

siguientes:

1. **Media aritmética**
   Es el algoritmo más sencillo que se va a implementar, básicamente se basa en realizar la media aritmética de los valores de temperatura leídos de los tres sensores. Este algoritmo tiene el problema que si falla alguno de los sensores, el valor de la media se distorsiona, alejándose mucho de la realidad, dando como resultado un valor incorrecto.

2. **Descarte de extremos**
   En este algoritmo se basa en lo siguiente:

   - Se leen los valores de los sensores y se almacenan en el conjunto E.
   - Se ordenan los valores del conjunto de menor a mayor.
   - Se eliminan los valores de ambos extremos (el mayor valor y el menor valor).
   - El valor restante es el resultado final.

   Con el presente algoritmo se soluciona el problema de fallo de uno de lo sensores, pero la medida obtenida es la de un único sensor por lo que no mejora la precisión.

3. **Convergencia rápida**
   Dado un conjunto E donde se encuentran los valores de temperatura leídos de los tres sensores y definida una distancia $\delta$ que indica la distancia máxima en la que un valor de temperatura puede considerarse aceptable respecto a otro valor. El algoritmo sería:

   - Se leen los valores de los sensores y se almacenan en el conjunto E.
   - Se colocan los valores aceptables (aquellos que cuya distancia con el resto sea menor que el $\delta$ definido) en un nuevo conjunto A.
   - Se calcula la media aritmética del conjunto de aceptados A.
   - Cualquier valor no aceptado que no esta en A se reemplaza por la media obtenida en el paso anterior.
   - Se calcula de nuevo la media aritmética de del conjunto modificado de A por el paso anterior y esta media será el resultado final.

   Con este algoritmo en caso de fallo de un sensor no se distorsiona la medida final por lo que se trata de un algoritmo robusto y tolerante al fallo de un sensor.

### Datos monitorizados

Se han de definir qué datos del equipo serán monitorizados por el periférico, estos datos tienen que ser representativos y útiles. Los datos que se pretenden monitorizar son los siguientes:

- **Fecha y hora del sistema**
  Se podrá ver la fecha y hora del sistema desde el periférico.

- **Espacio libre en disco**
  Se podrá consultar el espacio libre de la partición raíz (debido al espacio limitado de la pantalla se ha decido mostrar sólo el espacio de la unidad raíz).

- **Uptime**
  El tiempo que lleva encendida la máquina podrá consultarse desde el dispositivo.
- **Carga de la máquina**
  La carga media del sistema en el último minuto, los últimos 5 minutos y los últimos 15 minutos se podrá monitorizar desde el periférico.
- **Memoria RAM libre**
  La memora RAM libre expresada en unidades comprensibles por el usuario se mostrará en la pantalla del dispositivo cuando sea consultada.
- **Versión del kernel**
  En el dispositivo podrá consultarse la versión del kernel y la arquitectura para la que está compilado.
- **Dirección IP y MAC**
  La dirección IP y MAC de la tarjeta de red principal podrá ser consultada desde el periférico.
- **Temperatura de la CPU**
  La temperatura obtenida de los sensores de la CPU de la máquina cliente podrá ser consultada desde el dispositivo.

#### Acciones controladas

Una vez definidos los datos a monitorizar, y como el dispositivo es un periférico de monitorización y control, hay que definir las tareas a controlar. Las acciones que se podrán realizar desde el dispositivo son las siguientes:

- **Reiniciar el equipo**
  Se podrá apagar la máquina desde el dispositivo.
- **Apagar el equipo**
  Desde el periférico se podrá también apagar el ordenador controlado.

### 4.2.3. Diseño del demonio

Un **demonio (o daemon)** es un proceso no interactivo,que se ejecuta en segundo plano, y que no posee interacción con el usuario. Un demonio realiza tareas en tiempos predefinidos o en respuesta a algún evento del sistema y no hace uso de la entrada o salida estándar sino que usan logs o a otros demonios específicos para comunicar su estado.

La mayoría de los demonios son inicializados en el arranque del sistema y son hijos del proceso padre del sistema. Las acciones que se puede realizar sobre un demonio son las siguientes:

- **Iniciarlo**
- **Pararlo**
- **Reiniciarlo**

Se ha decido usar un demonio como cliente para el periférico debido a que el proceso debe ser iniciado al arrancar el sistema y permanecer invisible al usuario, así como permanecer en espera hasta que se conecte el periférico al puerto USB.

La estructura principal de un demonio UNIX es la siguiente:

- **Crear un proceso hijo (fork)**
- **Desasociar el proceso de una terminal tty**
- **Cambiar el directorio de trabajo**
- **Cambiar permisos**
- **Cerrar entrada y salida estándar**
- **Ejecutar bucle principal del demonio**

En el bucle principal del demonio se realizará la lógica que se muestra en el diagrama de flujo de la figura 25.

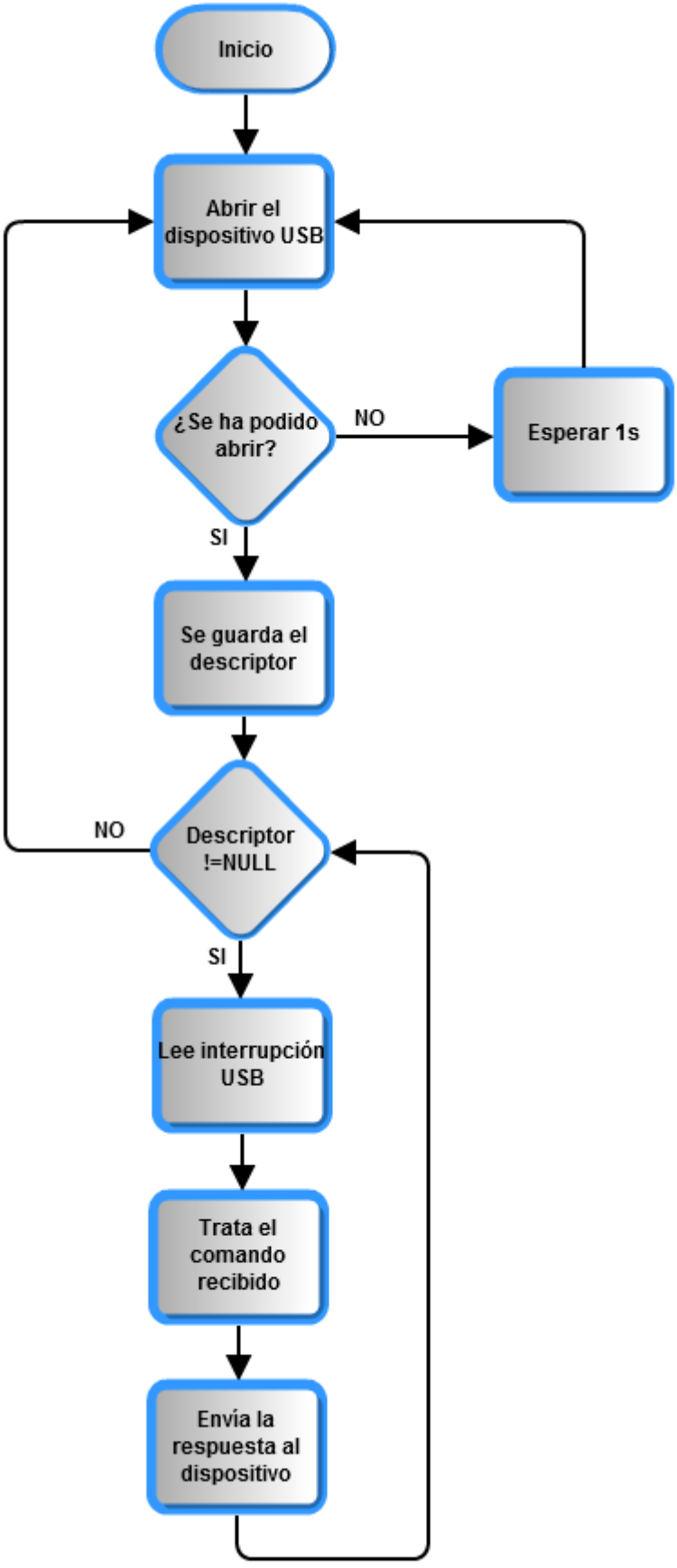


Figura 25: Diagrama de flujo del demonio

Una vez que ha comenzado el bucle principal, se intenta abrir el dispositivo USB, si no se encuentra (porque no está conectado), espera un segundo antes de volverlo a intentar, si lo encuentra en algún momento continua guardando el descriptor del dispositivo. Después si el descriptor no es nulo lee una interrupción del dispositivo por USB, guardando el valor leído.

Dependiendo del valor recibido del dispositivo se ejecuta el comando correspondiente a la petición enviando el resultado al periférico. Vuelve a repetir este proceso hasta que se desconecta el dispositivo, que regresa al comienzo del bucle principal.

El demonio guardará un log con la información y errores ya que no pueden hacer uso de la entrada y salida estándar del sistema.

Por otro lado, el demonio **monitorizará la temperatura leída** enviada por el periférico, cada un plazo determinado de tiempo, y la **escribirá en un log** en el disco duro del ordenador. El fichero de log será un fichero **CSV**, en el que se almacenará en cada linea la **fecha y hora de la lectura** así como el **valor leído** en grados centígrados, tal y como puede verse a continuación:

```
fecha hora,temperatura
```

# 5. Implementación

En esta sección se describirá la implementación del proyecto, tanto de la parte hardware como de la parte software.

## 5.1. Implementación hardware

En esta sección se comentará el desarrollo hardware en dos fases, la primera mediante un prototipo y la segunda mediante una placa de circuito impreso.

### 5.1.1. Prototipo

En primer lugar para comprobar el correcto funcionamiento del circuito, pudiendo realizar cambios en caso que se detecte algún error, y realizar la implementación software, se ha realizado un prototipo en una **breadboard** (o placa de prototipos). En la figura 26 puede observarse el prototipo del periférico montado sobre la breadboard.

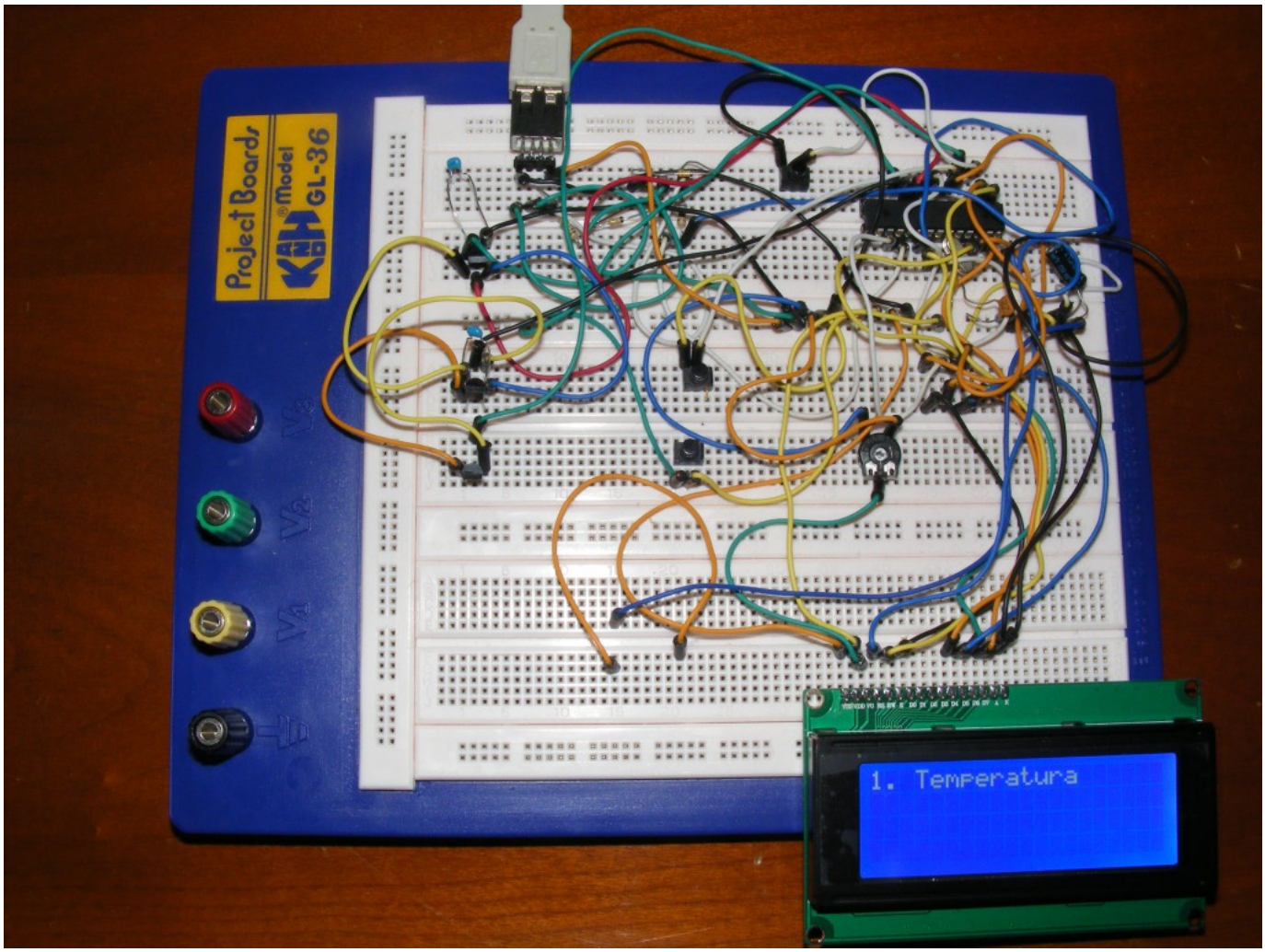


Figura 26: Prototipo del periférico

A continuación se procede a identificar cada parte montada en la protoboard, por bloques funcionales:

**Pantalla LCD**

Se puede observar en la siguiente figura la pantalla **LCD de 20x4** caracteres y la resistencia variable para el ajuste del contraste.

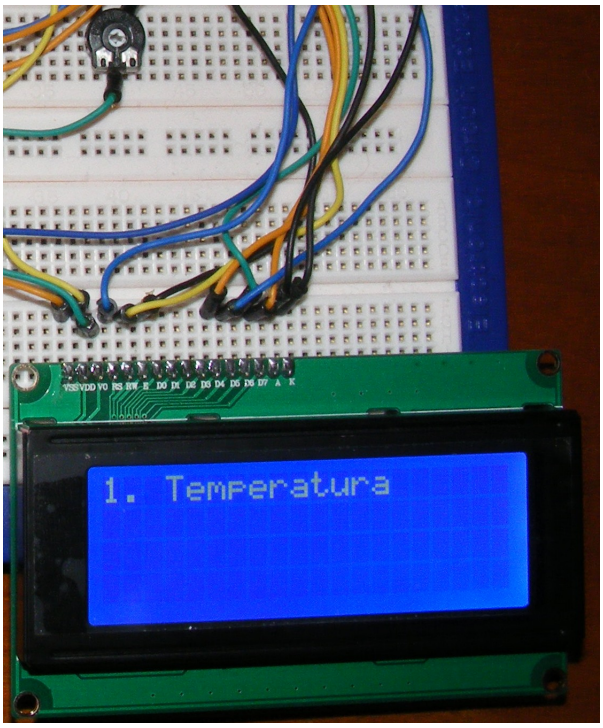


Figura 27: Prototipo: Pantalla LCD

**Microcontrolador**

Se puede ver el microcontrolador **Atmega168** así como su cristal de cuarzo y varios condensadores.

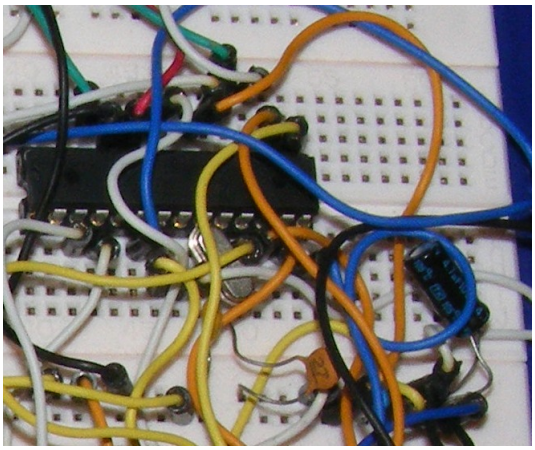

Figura 28: Prototipo: Microcontrolador

**Sensores de temperatura**

En la figura están los tres sensores de temperatura **LM35**.

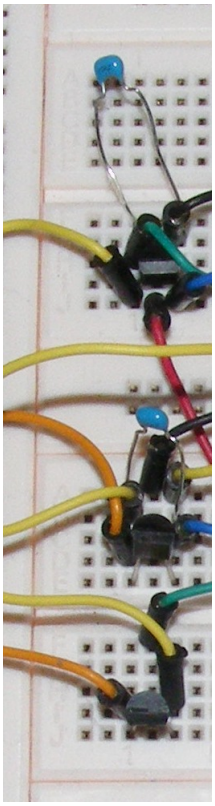

Figura 29: Prototipo: Sensores de temperatura

**USB**

Se puede localizar en la imagen el circuito necesario para el puerto USB, incluyendo además de las resistencias y diodos zener el conector USB hembra.

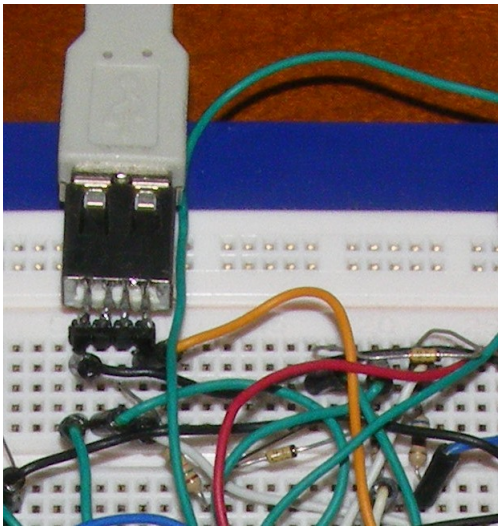

Figura 30: Prototipo: USB

**Pulsadores**

En la imagen se ven los tres pulsadores para controlar el menú, etiquetados con su función correspondiente.

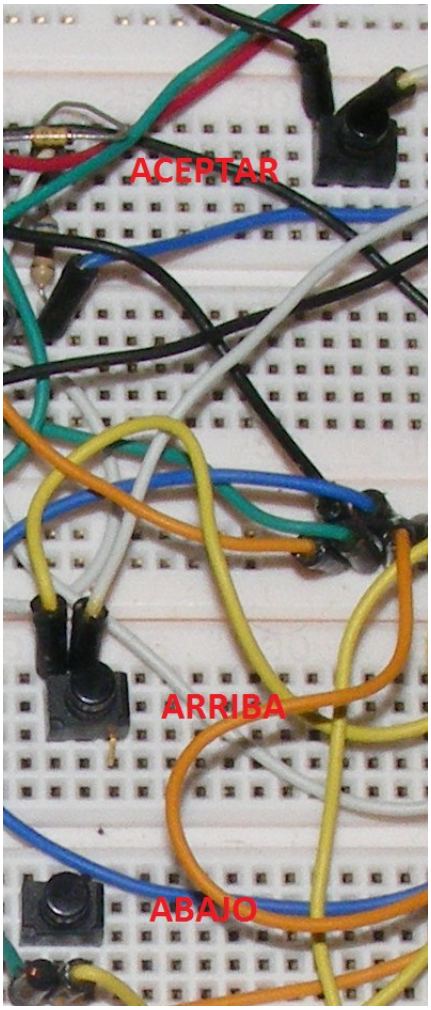


Figura 31: Prototipo: Pulsadores

### 5.1.2. Montaje de la placa de circuito impreso

Una vez que se ha recibido del fabricante la **PCB profesional**, se procede a montar el periférico. En la figura 32 se pueden ver una serie de herramientas que serán necesarias para su montaje, como un **soldador de 30W**, **estaño**, **pasta para soldar** y **alicates** de varios tipos(incluyendo un pelacables).


Figura 32: Herramientas necesarias para el montaje

En la figura 33 se pueden ver todos los componentes que forman el periférico incluyendo la placa de circuito impreso.

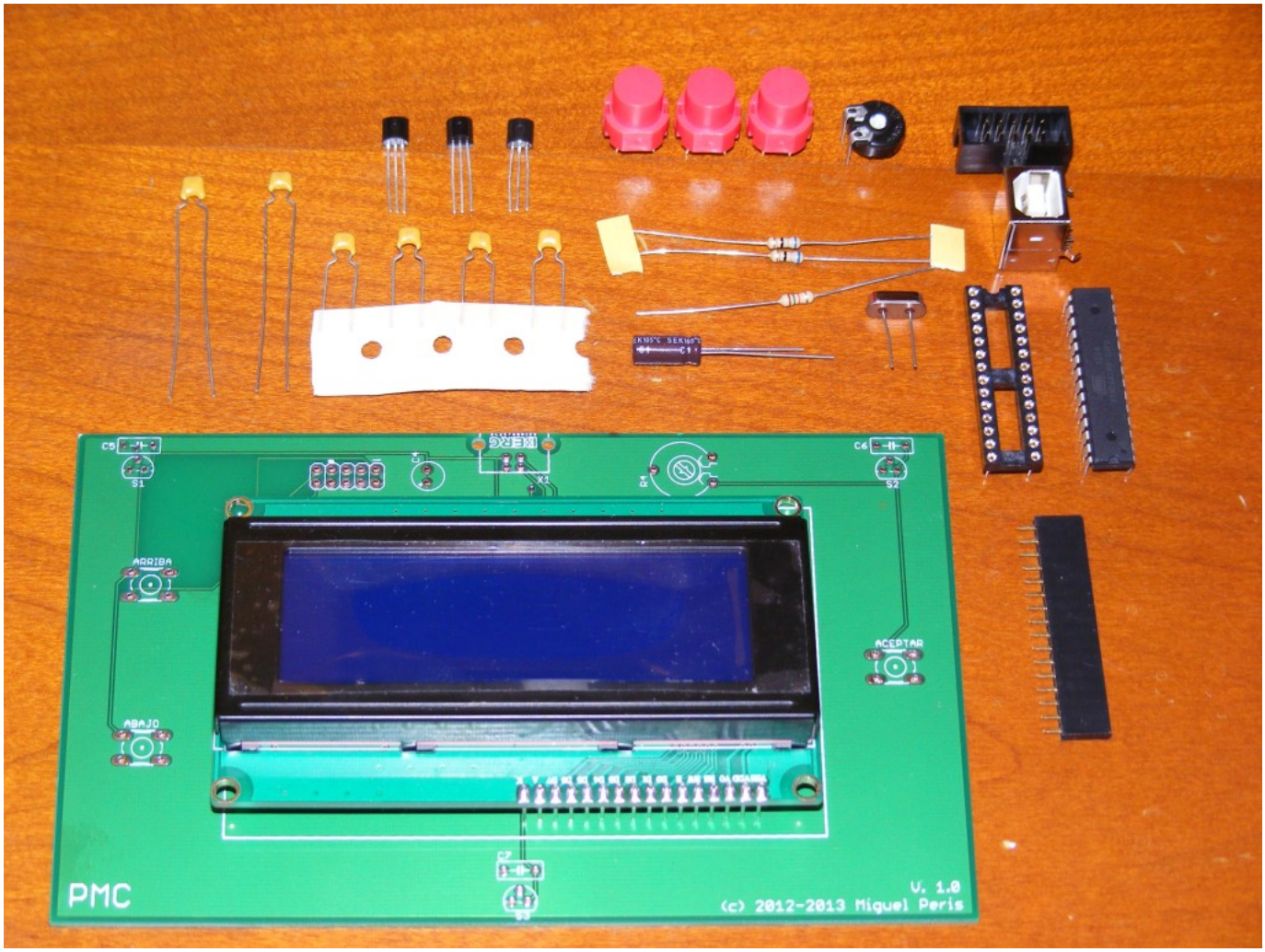


Figura 33: Componentes del periférico

Una vez que tenemos las herramientas y los materiales listos, se comenzará con el soldado de los componentes. Se irán soldando los componentes de menor a mayor altura, para que sea más fácil. En primer lugar se sueldan las resistencias y diodos (figura 34), a continuación se soldarán el cristal de cuarzo y el zócalo del microcontrolador (figura 35), después los condensadores cerámicos y electrolíticos (figura 36), más tarde se montan los sensores de temperatura, la resistencia variable, el conector ISP y el zócalo de la pantalla LCD (figura 37) y finalmente los pulsadores y el conector USB hembra (figura 38). Después de colocar los componentes, se irán cortando las patillas restantes con unos alicates de corte.

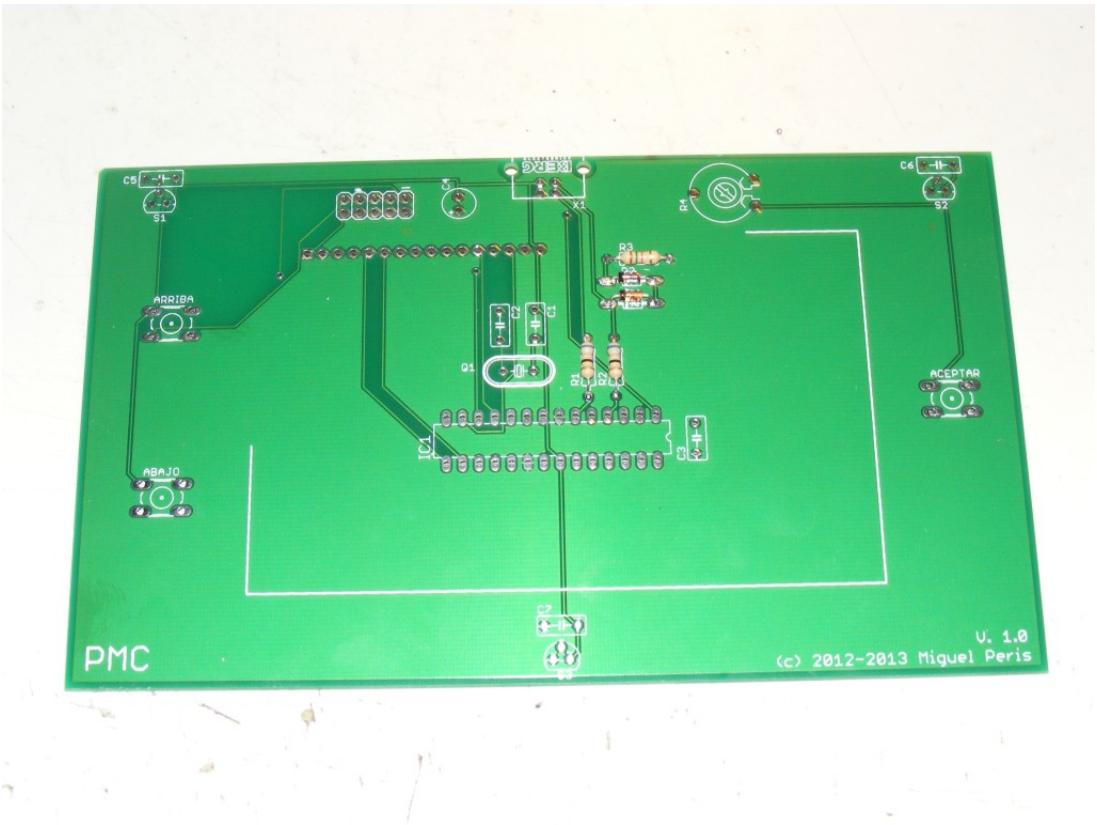


Figura 34: Montaje del periférico 1

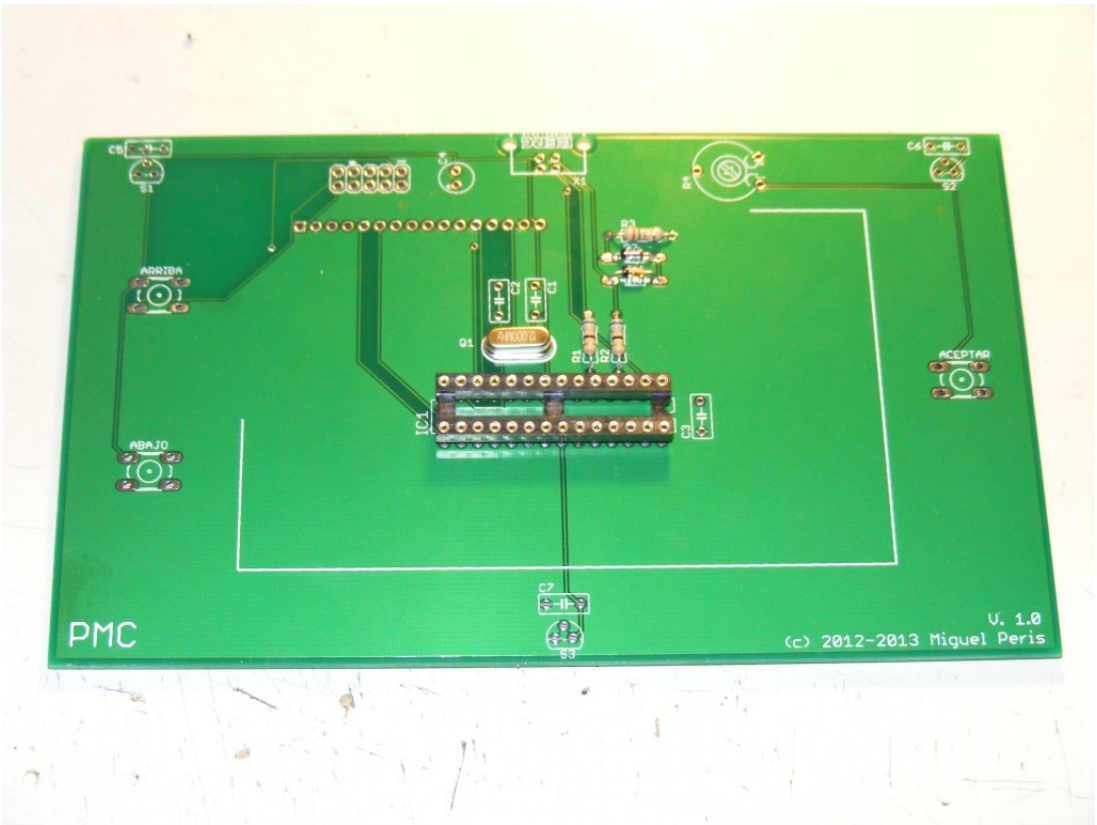


Figura 35: Montaje del periférico 2

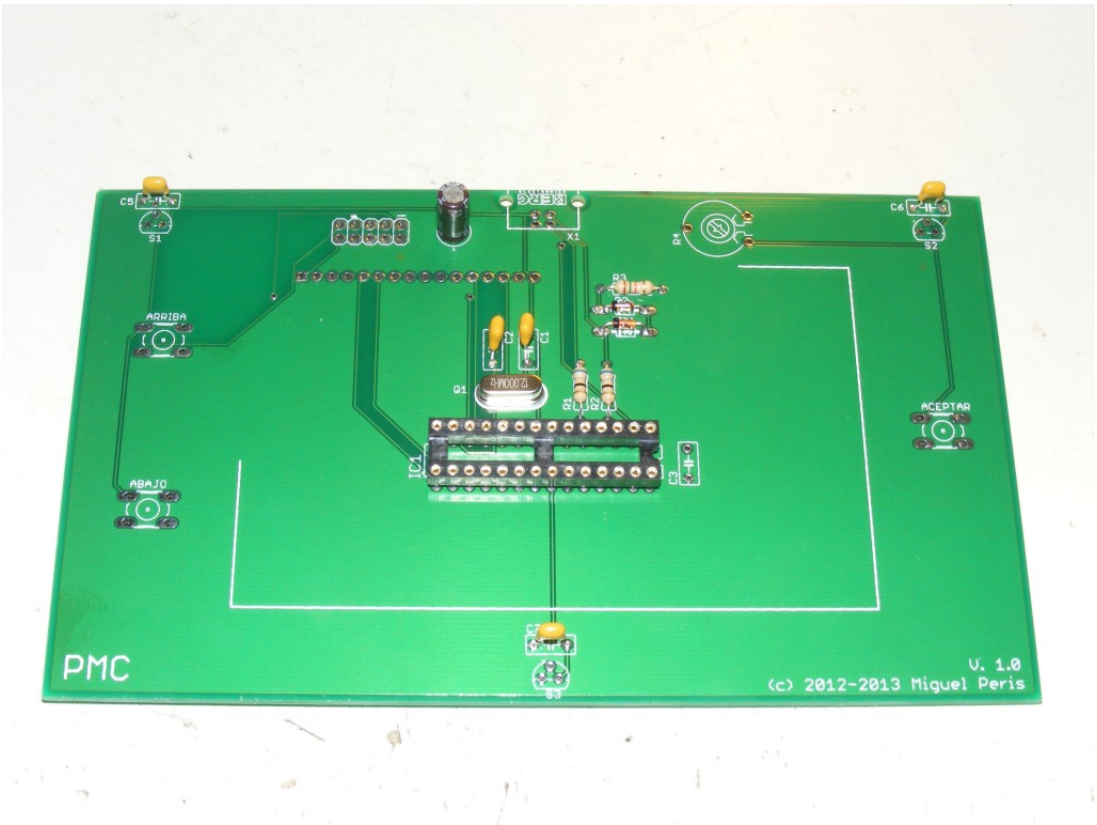


Figura 36: Montaje del periférico 3

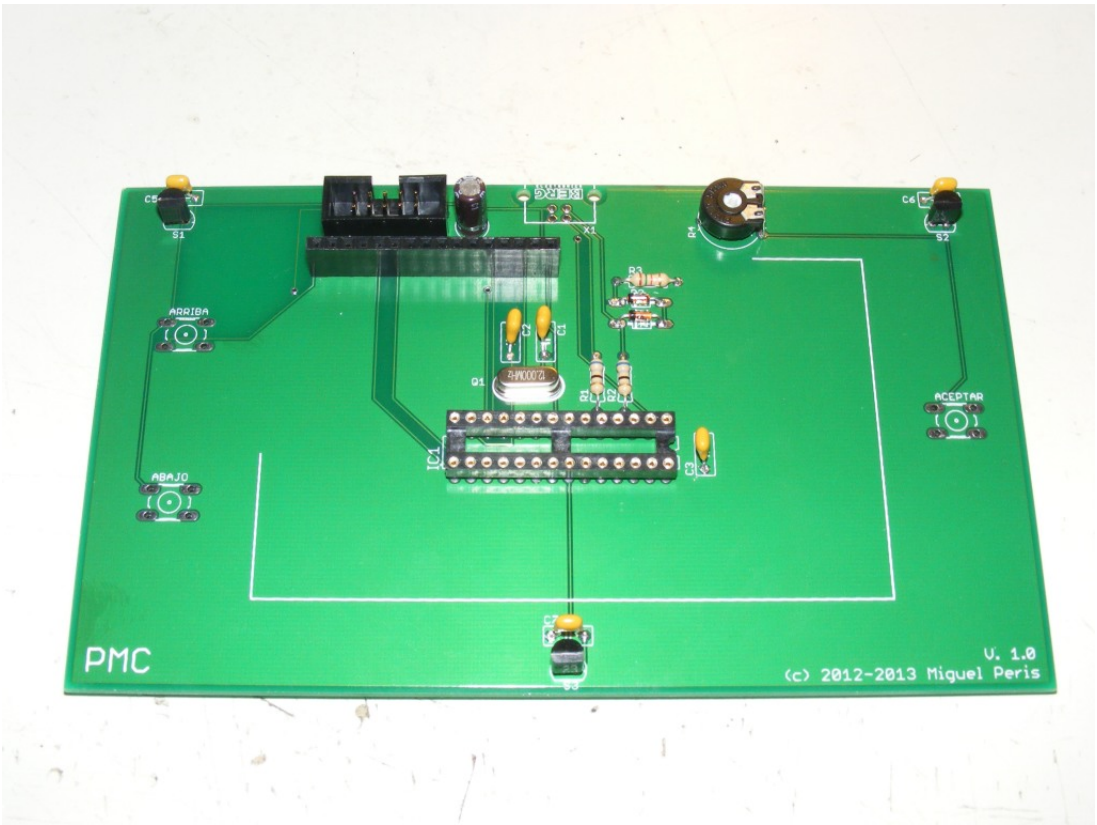


Figura 37: Montaje del periférico 4

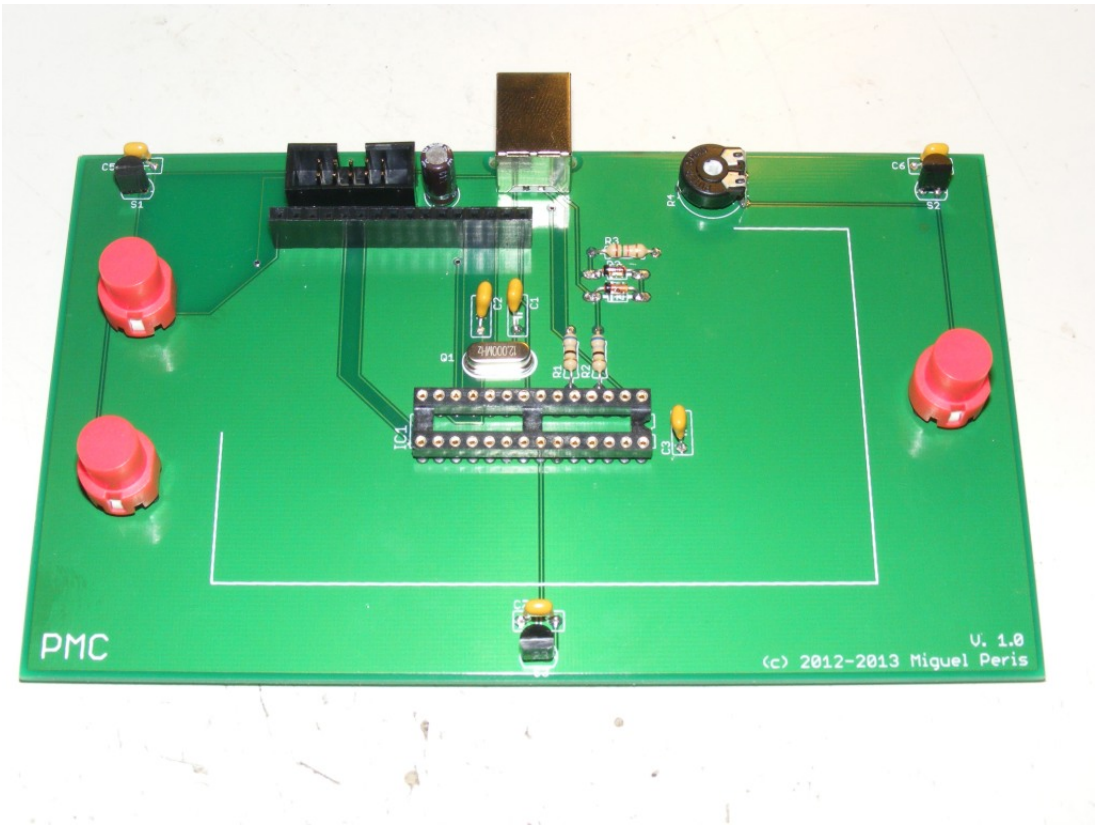


Figura 38: Montaje del periférico 5

Después de tener todo soldado y montado se realizará una tarea de comprobación de la alimentación antes de poner el microcontrolador para evitar que por algún problema de montaje o de la propia placa esté mal la alimentación. Para ello con un polímetro se medirá el voltaje entre las patillas 7 y 8 del zócalo del microcontrolador. Una vez verificado, se coloca el micrcontrolador en su zócalo, como puede verse en la figura 39.

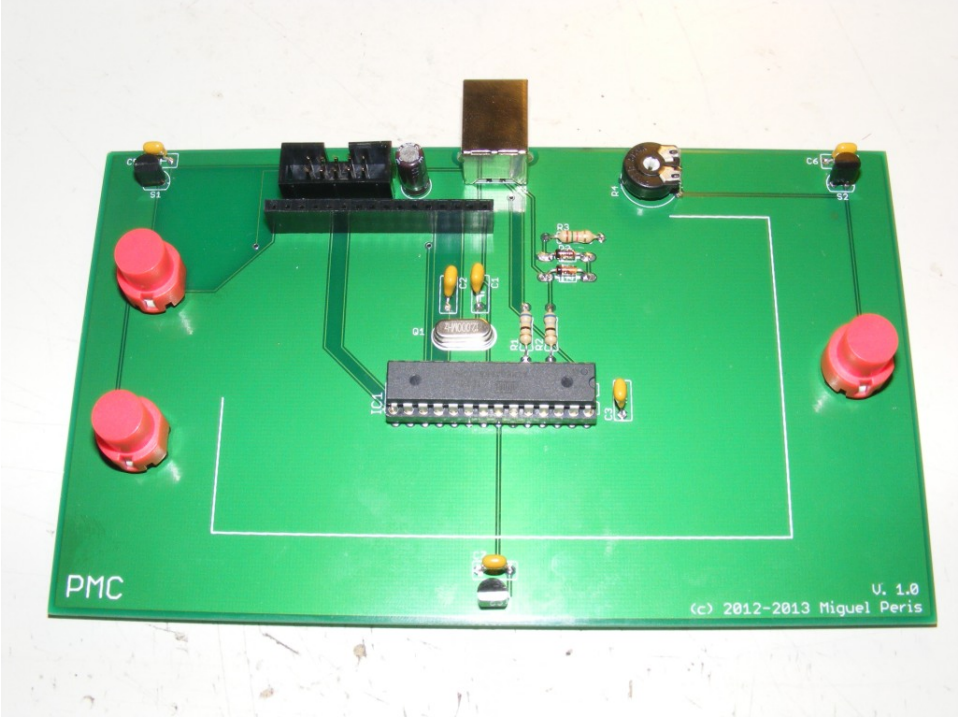


Figura 39: Montaje del periférico 6

Finalmente se coloca la pantalla LCD en su zócalo , tapando el microcontrolador, cristal de cuarzo y otros componentes dando un buen aspecto al periférico. Puede verse el resultado en la figura 40

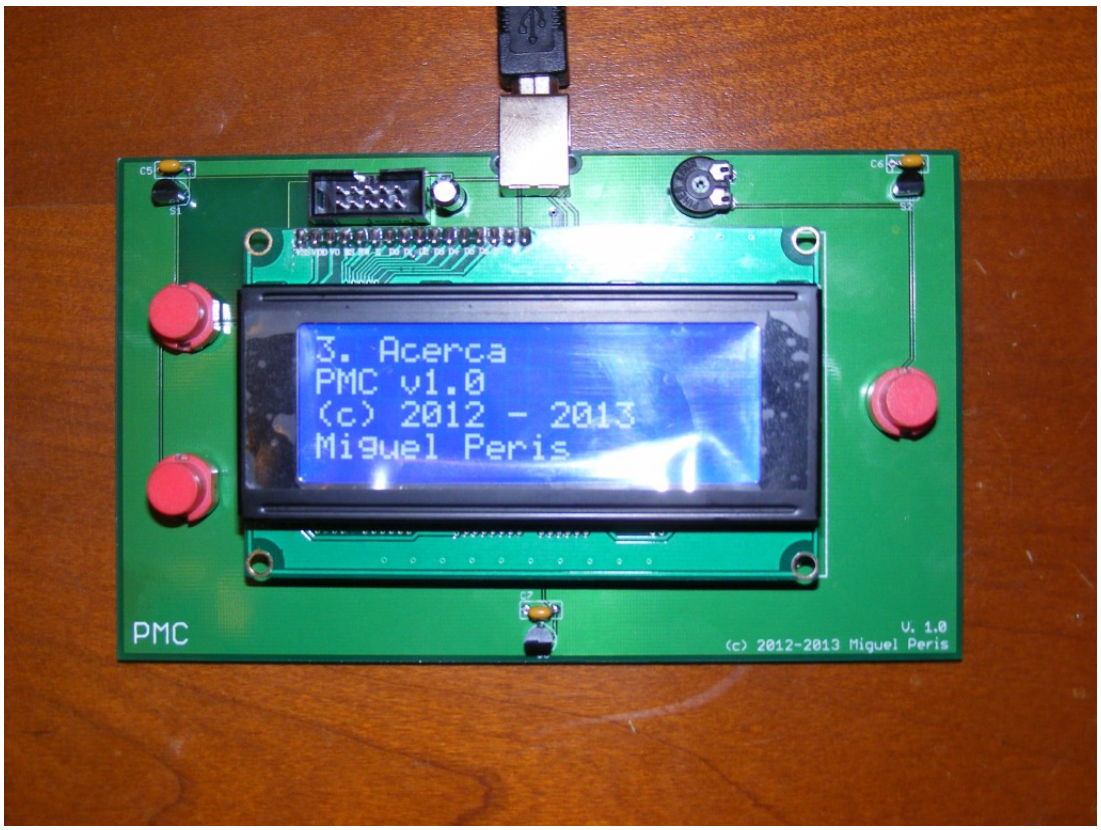


Figura 40: Periférico finalizado

**Ajuste del periférico**

Aparte de que es necesario tener grabado el firmware en el microcontrolador, sólo es necesario realizar un ajuste al periférico. Este ajuste es el **contraste** de la pantalla LCD que se controla con la resistencia variable **R4**. Para realizar este ajuste basta con alimentar el periférico y con un destornillador de precisión girar el tornillo de la resistencia variable hasta que la pantalla LCD se vea con el contraste deseado.

## 5.2. Implementación del firmware

El microcontrolador para funcionar necesita tener un firmware en su memoria de instrucciones, en el caso del Atmega168, memoria flash. El firmware ha sido programado en C basado en la librería libre **AVR Libc**[5] que se trata de una librería para programar el firmware de microntroladores de la familia AVR de Atmel.

Además de las librerías incluidas dentro de AVR Libc, se ha utilizado dos librerías externas, las cuales se pasan a detallar:

1. **V-USB**

   La librería V-USB[8] es el corazón de la comunicación USB del microcontrolador. La librería permite la implementación del protocolo USB vía software en microcontroladores AVR sin requerir integrados adicionales.

   V-USB implementa el estándar **USB 1.1** para dispositivos de baja velocidad necesitando un microcontrolador AVR de Atmel con al menos **2kb** de memoria **flash**, **128 bytes** de memoria **RAM** y una velocidad de reloj de al menos **12Mhz**.

2. **LCD.h**

   Liberia creada por Peter Fleury[9] para controlar pantallas LCD basadas en el controlador **HD44780** tanto en su modo de funcionamiento de **8 bits** como el de **4 bits**. Existen muchas librerías para controlar pantallas LCD, pero se ha escogido esta por las siguientes características:

   *a*) **Velocidad** (se probaron otras librerías que eran muy lentas).

   *b*) Capacidad de poder **cambiar libremente** todos los **pines de entrada/salida**.

   *c*) Consumo adecuado de memoria flash (aproximadamente **1.5k**).

Los sensores de temperatura **LM35** utilizados en el periférico tienen una salida lineal, donde cada **grado centígrado equivale a 10mV**. El microcontrolador leerá el voltaje que tiene el sensor en un instante determinado, por lo que para convertir el voltaje en grados bastará con **dividir el valor leído entre 2**.

La pantalla LCD utilizada es una pantalla de caracteres de **4 líneas de 20 caracteres** cada una. Debido a esto hay que optimizar en espacio, por lo que se ha decidido deja la primera linea de la pantalla (como se observa en la figura 41 marcado como *1*) para los menús, en esa línea se muestra el menú u opción en el que nos encontramos. Las tres líneas restantes

de la pantalla (marcado como *2* en la figura) se utilizan para el resultado de la opción seleccionada. Esta líneas se borrarán cuando se cambien de menú con los pulsadores de arriba/abajo.


Figura 41: Distribución en pantalla

También relacionado con la pantalla LCD, la librería utilizada permite mostrar caracteres de las tabla ASCII estándar, es decir los primeros 128 caracteres. Para mostrar la temperatura, es necesario mostrar su unidad, que en este caso son los **grados centígrados (ºC)**, pero el símbolo de grados (º) está incluido en la tabla ASCII extendida (del caracter 129 al 256), por lo que no se puede imprimir en la pantalla directamente. Para ello, la librería permite emplear **caracteres definidos por el usuario**, por lo que se ha decidido crear uno con el símbolo del grado.

Para especificar este caracter se debe definir una **matriz 8x8 bits** donde cada bit es un pixel. Un **1** en cualquier posición de la matriz indica que el pixel está activo y un **0** indica que el pixel no está activo. Se muestra a continuación dicha matriz.

```
const unsigned char grados[8] PROGMEM=
{
0b00000111,
0b00000101,
0b00000111,
0b00000000,
0b00000000,
0b00000000,
0b00000000,
0b00000000
};
```

Los menús están alojados también en el firmware del microcontrolador, y se han definido

unos menús con dos niveles de profundidad. Los menús definidos son los siguientes:

1. **Temperatura**
   Este menú engloba todas las acciones relacionadas con los sensores de temperatura presenten en el periférico. Contiene las siguientes opciones:

   *a*) **Todos sensores**
   Se muestra la lectura de los los tres sensores de temperatura en el momento a acceder a él.

   *b*) **Votación**
   En este submenú se muestran los resultados de los tres algoritmos de votación implementados.

2. **Monitorización**
   Aquí se agrupan todas las tareas encargadas de la monitorización de datos del PC por parte del periférico. Los opciones que lo componen son las siguientes:

   *a*) **Fecha y hora**
   Se muestra la fecha y hora del PC.

   *b*) **Espacio en disco**
   Se obtiene el tamaño, el espacio ocupado y el espacio disponible en la partición raiz del disco duro del PC monitorizado.

   *c*) **Uptime y carga**
   Presenta el uptime y la carga momentánea del PC.

   *d*) **Memoria libre**
   Muestra la cantidad total de memoria RAM, la cantidad usada y la cantidad libre expresada en Megabytes.

   *e*) **Kernel**
   Se consulta la versión y arquitectura del kernel del PC.

   *f*) **Red**
   Se expone la dirección MAC de la máquina así como la dirección IP.

   *g*) **Temperatura cpu**
   En este submenu se muestra la temperatura de la CPU de la máquina monitorizada.

3. **Control**
   En el menú Control se agrupan las acciones de control que puede realizar el periférico sobre el PC. Dichas acciones son las siguientes:

   *a*) **Reiniciar equipo**
   Permite reiniciar la máquina conectada al dispositivo.

   *b*) **Apagar equipo**
   Desde esta opción se podrá apagar el ordenador.

4. **Acerca**
   En este menú simplemente se muestra la información acerca del dispositivo, incluyendo la versión y el copyright.

A continuación se muestran las imágenes de los menús y opciones implementados en la pantalla LCD del dispositivo:

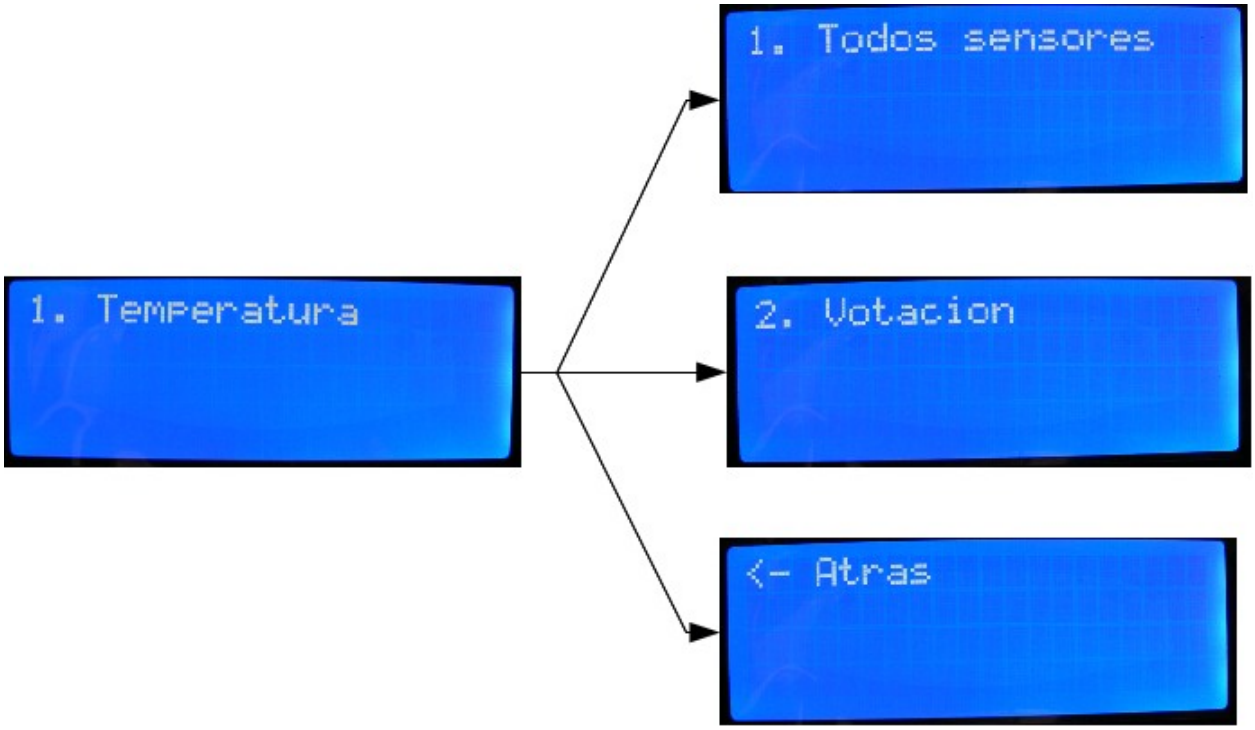


Figura 42: Menús implementados: Temperatura

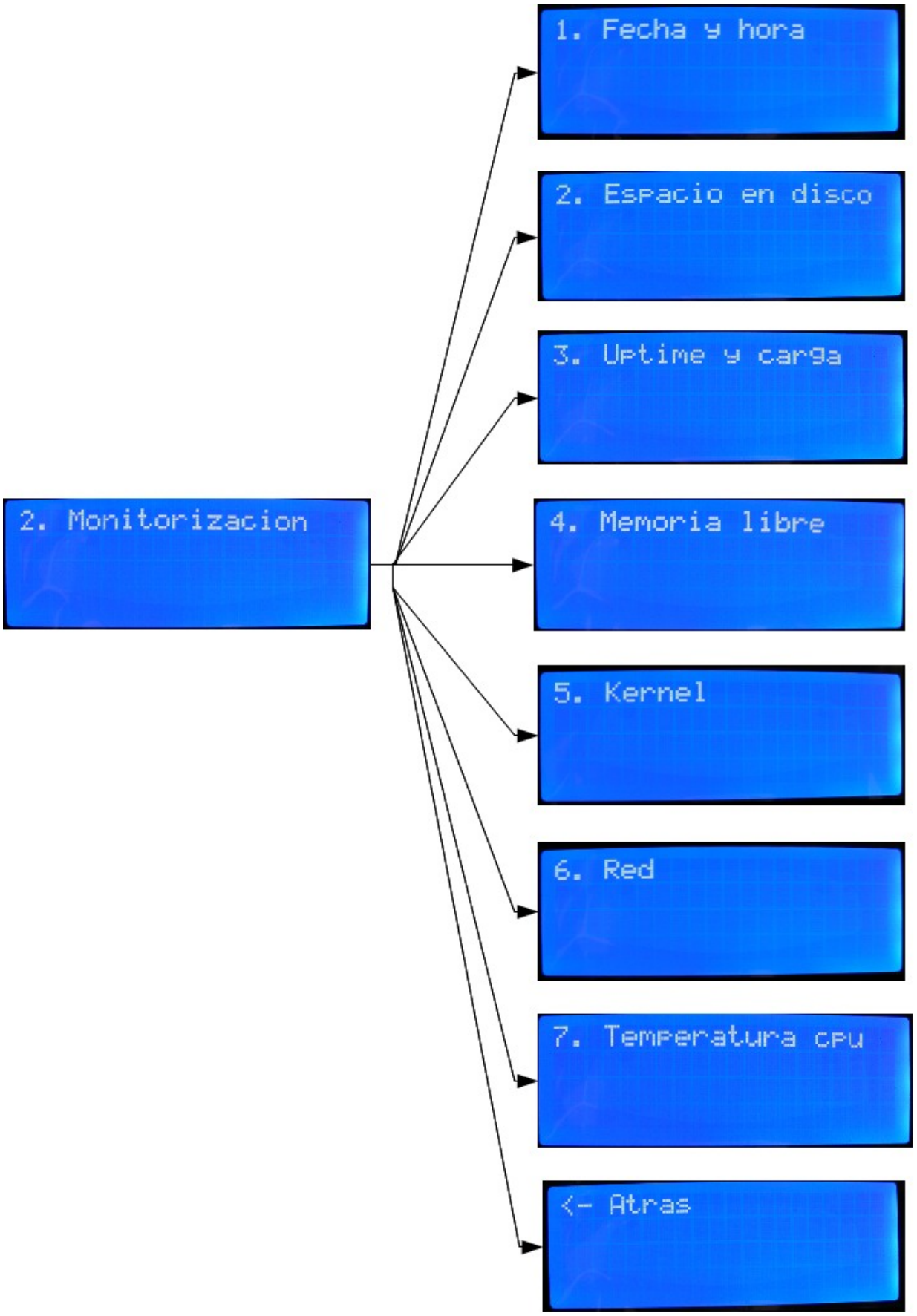


Figura 43: Menús implementados: Monitorización

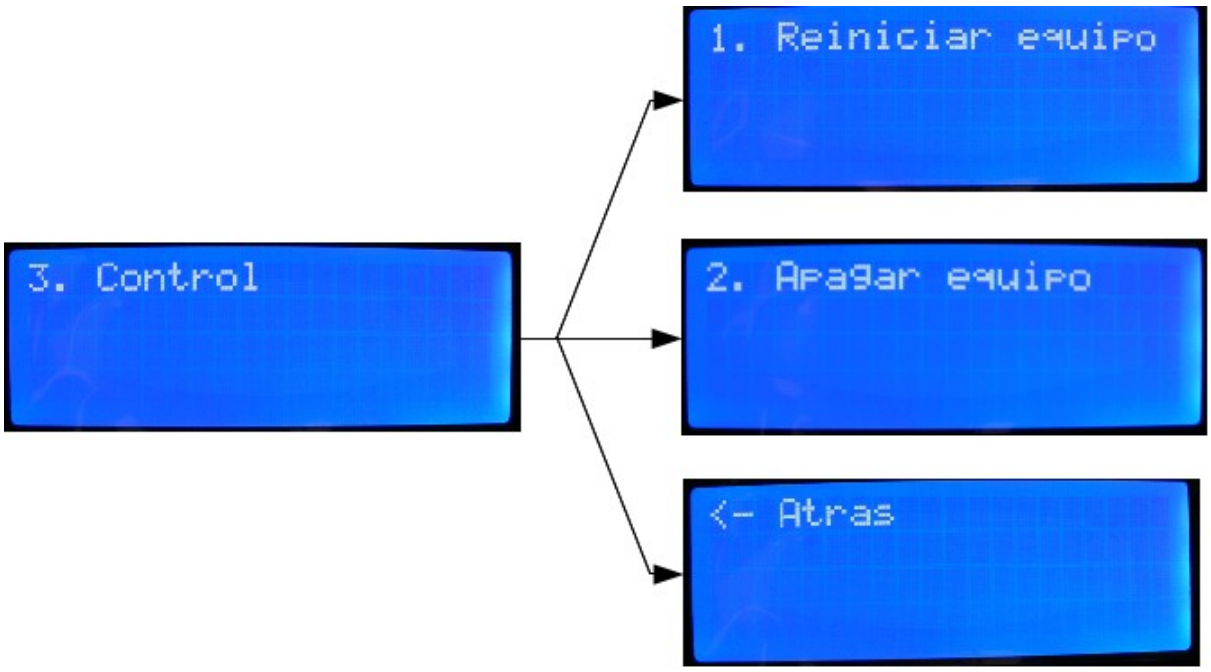


Figura 44: Menús implementados: Control

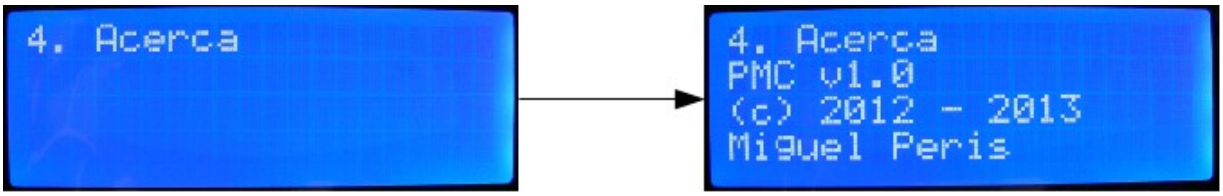


Figura 45: Menús implementados: Acerca

Para realizar las peticiones de información desde el dispositivo al ordenador se ha definido una serie de comandos basados en un carácter, que posteriormente serán tratados por el demonio y preparará la respuesta correspondiente al periférico. La serie de comandos que pueden ser enviados por el dispositivo al ordenador y su correspondiente petición se detalla en la tabla 37.

| **Comando** | **Petición** |
|---|---|
| **f** | Fecha y hora del sistema |
| **d** | Espacio libre en disco |
| **u** | Uptime y carga |
| **m** | Memoria RAM libre |
| **k** | Versión del kernel |
| **r** | Dirección IP y MAC |
| **T** | Temperatura de la CPU |

Tabla 37: Comandos enviados desde el dispositivo

Respecto a los comandos de control remoto, los dos únicos que hay se muestran en la tabla 38.

| Comando | Petición |
|---|---|
| **R** | Reinicio de la máquina |
| **A** | Apagado de la máquina |

Tabla 38: Comandos de control enviados desde el dispositivo

Estos comandos son enviados de manera **no bloqueante** al PC conectado mediante el establecimiento de una interrupción. El resultado será enviado por parte del PC como otra interrupción que será tratada por el microcontrolador escribiendo los datos recibidos en la parte correspondiente de la pantalla LCD.

Gracias a la librería V-USB, cuando se recibe una interrupción por parte ordenador, es tratada por una función distinta si es de lectura o de escritura. Para la de escritura, cuando se recibe un buffer de datos correspondiente a una petición realizada anteriormente por el dispositivo, se va escribiendo en la parte de la pantalla LCD correspondiente (las tres últimas líneas), realizando cambio de líneas cuando se llena una.

Finalmente también cabe destacar que ante una petición por parte del demonio de la temperatura se enviará el valor resultante del algoritmo de **convergencia rápida**.

## 5.3. Implementación del demonio

El demonio ha sido implementado en C basándose en las liberas **POSIX** de UNIX además de usar la librería **libusb**[6] para la comunicación por USB.

La librería libusb permite a las aplicaciones que la usan un fácil acceso a dispositivos USB permitiendo intercambiar mensajes y datos. No está disponibles únicamente para C/C++, sino que también lo está para una decena de lenguajes de programación como Java, Python, C#, Perl, Ruby, etc. A sí mismo, libusb es **multiplataforma** estando disponible sistemas operativos Unix-like(Linux, BSD, MacOS X) y Windows.

A la hora de implementar el demonio se ha seguido la estructura básica de un demonio genérico definida en el apartado de *Diseño del sistema*. Además de esto, para poder parar el demonio es necesario saber el **PID del proceso**. Para solventar eso, en los demonios habitualmente se guarda el PID al ejecutar el proceso en un **fichero .pid** ubicado en ***/var/run/***. Dicho fichero se crea cuando arranca el demonio y se borra cuando se para, además para evitar que haya más de una instancia del demonio ejecutándose se verifica la existencia del fichero en el arranque, y se existe se interrumpe la ejecución de la nueva instancia.

El demonio, para contestar las peticiones del periférico(identificadas por un carácter) ejecutará una serie de comandos UNIX que obtendrán los datos del sistema. Estos comandos son genéricos en casi cualquier sistema UNIX y sin necesidad de tener algún paquete no esencial(salvo el caso de la temperatura de la CPU). Los comandos usados por cada petición son los siguientes:

- **Fecha y hora del sistema:**

  ```
  date +'%H:%M:%S %d/%m/%Y'
  ```

  La fecha y hora del sistema se obtendrá en una sola línea con el formato HH:MM:SS DD/MM/AA.

- **Espacio libre en disco:**

  ```
  df -h / |tail -n1|tr -s " " " "
  ```

  El espacio en disco será obtenido en unidades legibles (dependiendo del tamaño del disco, pero en general en Gb.), mostrándose el espacio total de la partición, el espacio ocupado, el espacio disponible y el porcentaje de uso.

- **Uptime:**

  ```
  uptime | awk '{sub(/\,/,"");print $3" " $4 }'
  ```

  Se muestra con este comando el tiempo que lleva encendida la máquina.

- **Carga de la máquina:**

  ```
  uptime |cut -f3- -d,|cut -f2 -d:
  ```

  Muestra la carga media del sistema en el último minuto, los últimos 5 minutos y los últimos 15 minutos.

- **Memoria RAM libre:**

  ```
  free -m | tr -s " " " "| cut -f2-4 -d" "|head -n2|tail -n1
  ```

  Se muestra la memoria RAM total, usada y libre expresada en Megabytes.

- **Versión del kernel:**

  ```
  uname -rm
  ```

  El comando obtiene la versión del kernel en ejecución así como la arquitectura del mismo.

- **Dirección IP y MAC:**

  ```
  IP: ifconfig eth0 | grep 'inet addr:' | cut -d: -f2 | awk '{ print $1}'

  MAC: ip link show eth0 | awk '/ether/ {print $2}
  ```

  Se obtienen las direcciones IP y MAC de la primera tarjeta de red (eth0).

- **Temperatura de la CPU:**

  ```
  vcgencmd measure_temp
  ```

Se consulta la temperatura a la que está en ºC la CPU. En este caso se utiliza un comando específico para el sistema donde se realizarán las pruebas.

Para realizar las acciones de control, también se utilizarán comando UNIX genéricos, dichos comandos son los siguientes:

- **Reiniciar el equipo:**

  `reboot`

  Se reinicia el equipo de manera adecuada.

- **Apagar el equipo:**

  `halt`

  Se iniciará el apagado del sistema con este comando.

El resultado de estos comandos es almacenado en un buffer y es enviado al periférico enviado un mensaje de control.

Los mensajes de error y de información serán guardados en un log (dado que las salida estándar y de error están cerradas). Dicho log es el log usado habitualmente por los demonios y es el **daemon.log** ubicado en ***/var/log***.

La función de **historificación de la temperatura** recibida del periférico se realizará **cada 30 segundos**. Se almacena en el fichero **PMChist.log** ubicado en la ruta **/var/log** con el formato que se ha definido en el *Diseño del demonio*.

Aparte de desarrollar el ejecutable del demonio es necesario realizar un shell script que servirá para arrancar, parar y reiniciar el demonio. Dicho script en sistemas UNIX se ha de ubicar en la ruta ***/etc/init.d***.

El script se basa en un *case* que dependiendo de del parámetro recibido (start, stop, restart) arrancará, parará o reiniciará el demonio. Además lleva una cabecera como la que se muestra a continuación:

```
### BEGIN INIT INFO
# Provides:          scriptname
# Required-Start:    $remote_fs $syslog
# Required-Stop:     $remote_fs $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start daemon at boot time
# Description:       Enable service provided by daemon.
### END INIT INFO
```

En esta cabecera se especifica el nombre del script, los requisitos para arrancar y parar el demonio, los runlevel en los que se iniciará el demonio, los runlevel en los que se detendrá el demonio, así como una descripción corta y una descripción larga del demonio. A este script habría que darle **permisos 755** así como que su propietario y grupo sea **root**.

Con el script en */etc/init.d* se puede iniciar y para de forma manual, pero para que se inicie al arrancar el sistema es necesario crear enlaces simbólicos en los directorios ***/etc/rcX.d*** dónde X es el runlevel. Dentro habría que realizar dos enlaces, dependiendo de la acción que se requiera realizar, uno con *S<num><nombreScript>* y otro *K<num><nombreScript >*. El que comienza por S indica que iniciará el demonio y el que comienza por K indica que parará el demonio, al cambiar al runlevel donde se realice el enlace.

Estos enlaces se puede crear de manera manual, o se pueden crear mediante el comando **update-rc.d** que creará los enlaces según los runlevel de arranque y parada indicados en el script ubicado en init.d. El comando completo utilizado es el siguiente:

```
update-rc.d PMC defaults 21
```

Dónde **21** es el número de secuencia que indica en que orden se ejecutarán todos los scripts de arranque. Se ha de usar un número de secuencia que no esté ocupado por otro script. Por otro lado, PMC es el nombre del script que hemos creado en init.d.

Con esto el demonio ya estaría configurado para que se inicie cada vez que el equipo arranque.

# 6. Pruebas

En este capítulo se detallarán las pruebas realizadas para comprobar el correcto funcionamiento y su cumplimiento con las especificaciones del dispositivo.

## 6.1. Pruebas unitarias

En esta subsección se realizarán pruebas sobre el periférico de manera aislada, sin conectarlo al ordenador por USB, para comprobar el correcto funcionamiento del mismo. Para alimentarlo (ya que se alimenta únicamente por el puerto USB) y evitar comunicación con el PC, se conectar a un alimentador USB de teléfono móvil.

En primer lugar se realizarán pruebas de funcionamiento básicas, como manejo de menús, pulsadores, etc., para posteriormente probar algoritmos de votación.

#### Pruebas de funcionamiento básico

Con estas pruebas unitarias se pretende comprobar el funcionamiento básico del periférico (de manera aislada). Se realizan las siguientes pruebas:

1. **Funcionamiento de la pantalla LCD**
   Se alimentará el dispositivo por USB y se verificará que en la pantalla LCD sale el primer menú (Temperatura) y se ve correctamente. En caso de no verse bien se ajustaría el contraste como se explica al final del apartado 5.1.1.

2. **Funcionamiento de los pulsadores y menús**
   Se comprobará el correcto funcionamiento de los menús y el manejo de los mismos con los pulsadores. Se usarán los pulsadores de Arriba y Abajo comprobando que cambia al menú superior e inferior respectivamente en la pantalla LCD y se usará el de Aceptar para entrar en el submenú u opción mostrada actualmente.

3. **Lectura de temperatura de los sensores**
   Se realizará una prueba sencilla para validar el correcto funcionamiento de los sensores de temperatura, que consiste en acceder al menú ***Temperatura/Todos sensores*** del periférico (como se puede observar en la figura 46) viendo que los valores arrojados son aceptables.


Figura 46: Temperatura de los tres sensores

### Pruebas de los algoritmos de votación

Para realizar las pruebas de los algoritmos de votación se utilizó el prototipo en la bread-board, para sí poder desconectar los sensores de temperatura.

#### Prueba de tolerancia a fallos

Para comprobar la robustez de los algoritmos de votación implementados en el periférico, se realizara un prueba simulando el fallo de sensores. Los pasos que se han realizado en la prueba son los siguientes:

1. **Lectura de los valores de temperatura de todos los sensores**
   Como se puede observar en la figura 47 se ha procedido a consultar lo valores de cada uno de los tres sensores de temperatura para comprobar que están dado valores correctos y para tener un punto de referencia.

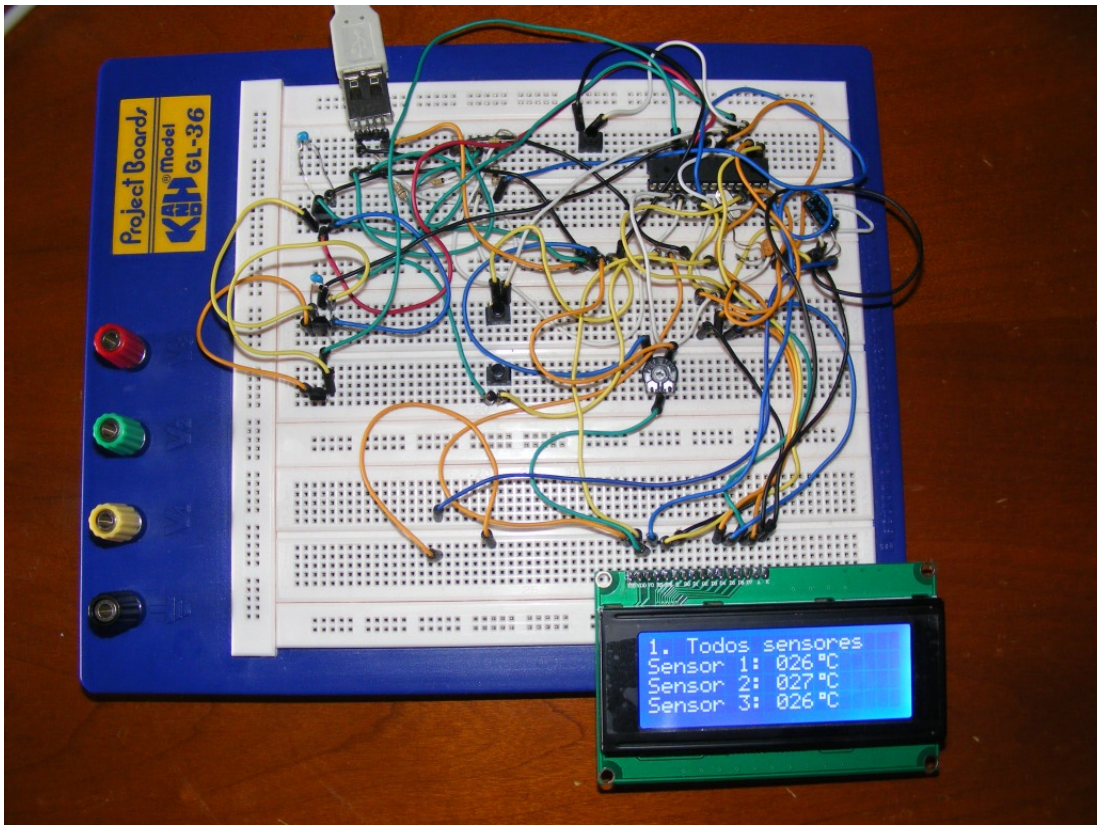


Figura 47: Lectura de la temperatura de los tres sensores

2. **Resultados de los algoritmos de votación**
   Posteriormente, se obtuvieron los resultados de los tres algoritmos de votación, la media aritmética, descarte de extremos y convergencia rápida, como pueden observarse en la figura 48. Los valores obtenidos son correctos, aunque la medida del valor de descarte de extremos hace indicar que bajó la temperatura con respecto al punto anterior.

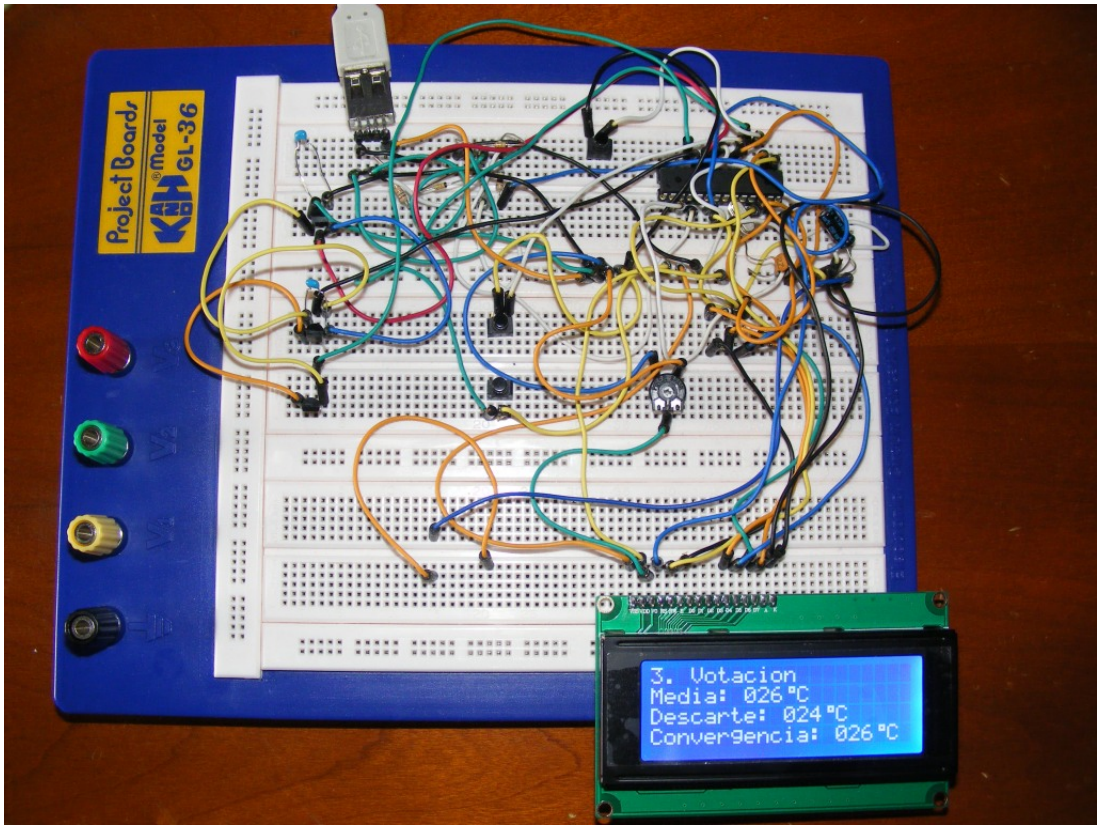


Figura 48: Resultados de los algoritmos de votación

3. **Fallo de un sensor**
   Se procedió a simular el fallo de un sensores temperatura desconectándolo de la placa de prototipos, como se aprecia en la imagen de la figura 49. Se volvió a pedir el cálculo de los algoritmos de votación y como se puede observar, la media aritmética de aun valor erróneo, debido a desconectar una sensor la temperatura leída por el microcontrolador es un valor aleatorio, habitualmente alto. Por otro lado se ve como en los valores de los algoritmos de descarte de extremos y convergencia rápida dan medidas aceptables.

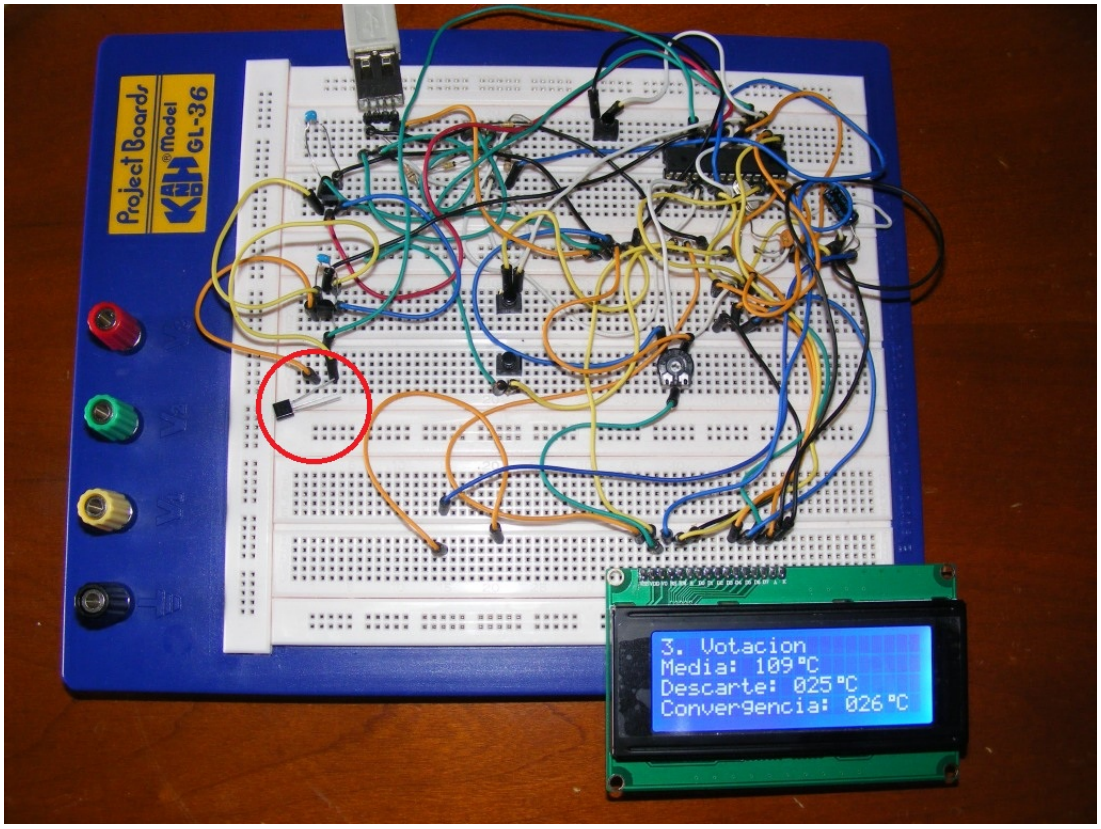


Figura 49: Fallo de un sensor

Se puede comprobar que los algoritmos de descarte de extremos y convergencia rápida **son robustos y tolerantes al fallo de un sensor**.

4. **Conexión de nuevo del sensor**
   Se procedió a conectar de nuevo el sensor de temperatura a la placa y se mostraron de nuevo los resultados de los tres algoritmos de votación. Como puede apreciarse en la figura 50 se vuelven a obtener valores aceptables en los tres algoritmos.

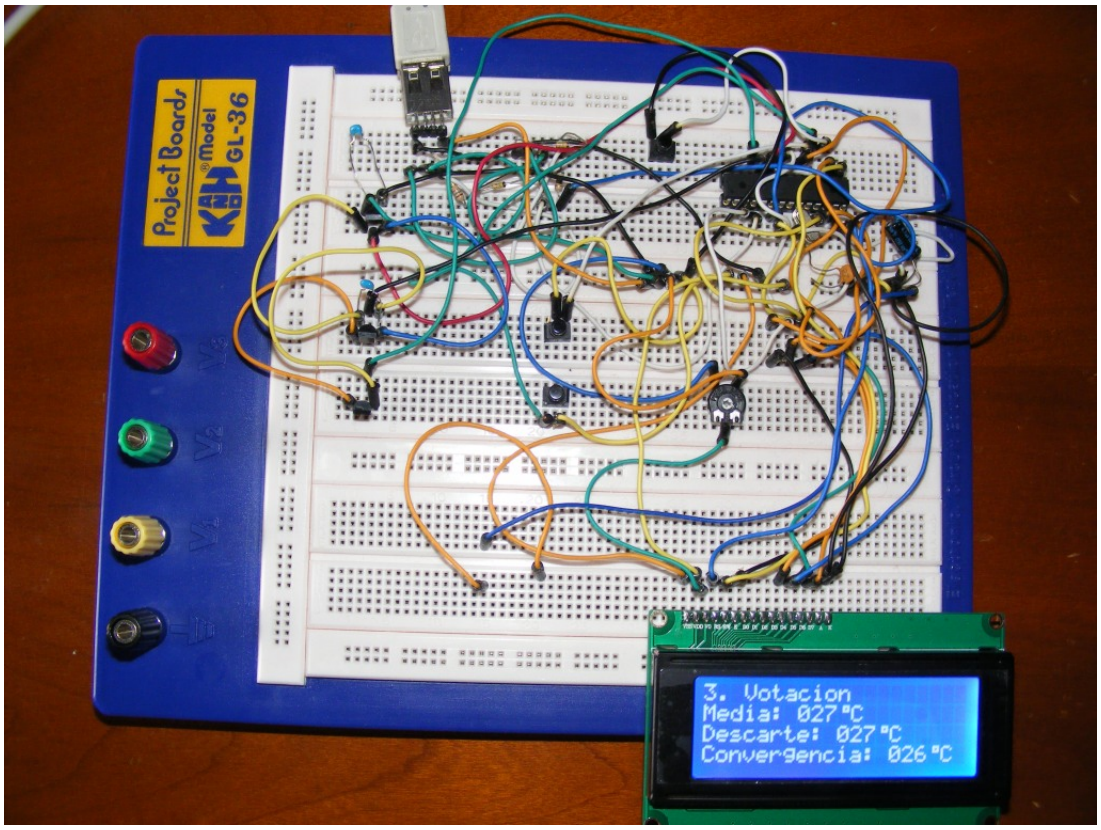


Figura 50: Conexión de nuevo del sensor

5. **Fallo de dos sensores** Pese a que a priori se sabe que ningún algoritmo de los implementado es tolerante al fallo de dos de los tres sensores de temperatura, se ha realizado la prueba para ver los resultados. Dichos resultados pueden observarse en la figura 51

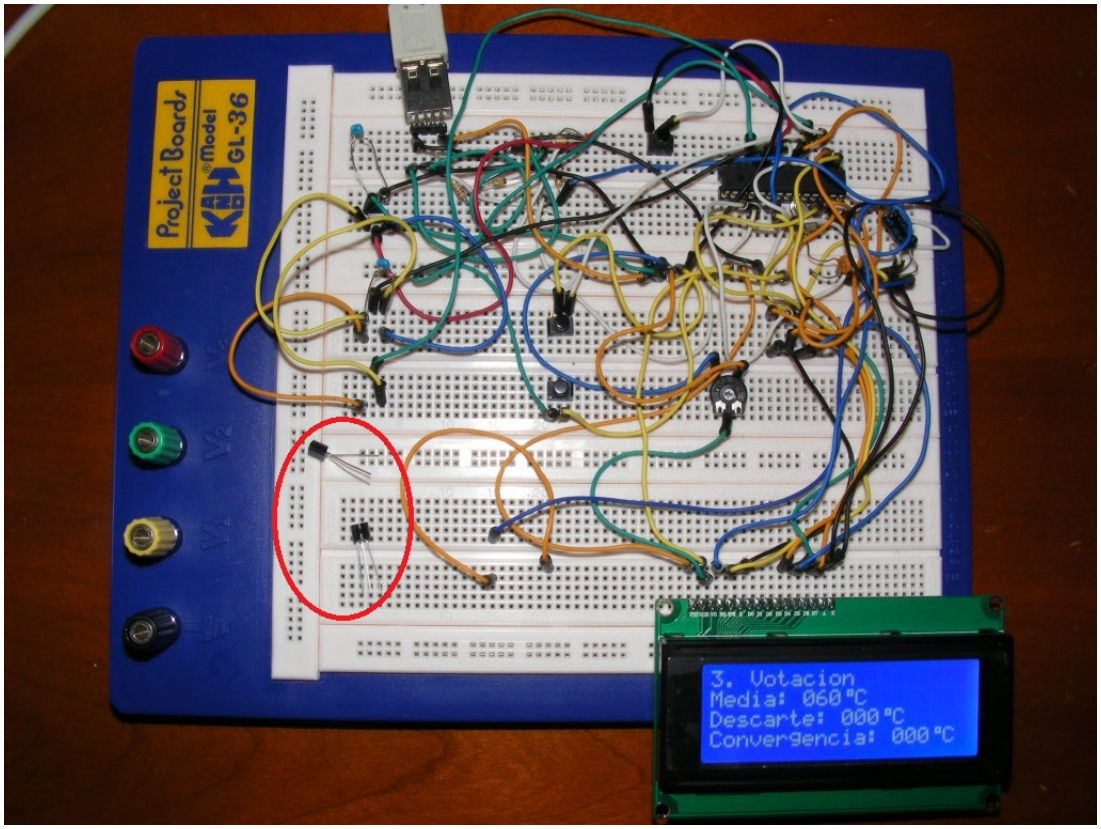


Figura 51: Fallo de dos sensores

## 6.2. Pruebas integradas

Este apartado trata sobre las pruebas integradas realizadas, es decir, las pruebas del sistema completo, incluyendo la comunicación del dispositivo con el ordenador.

Para realizar las pruebas integradas no se va a usar un ordenador tradicional, sino que se va a usar una **Raspberry pi**. La Raspberry pi (modelo B, que es el que se usará) es un miniordenador SBC se bajo coste desarrollado en Reino Unido, basado en un **System-o-a-chip** BCM2835 que incluye una **CPU ARM** a **700Mhz.**, una **GPU** y **256 o 512 Mb.** de memoria **RAM**.

Su tamaño es únicamente de 85.60mm x 56mm x 21mm y pesa 45g. Dispone de conexiones **HDMI** y **vídeo compuesto** para el vídeo, jack de 3.5mm para el audio, 2 puertos **USB 2.0** y una **tarjeta de red 10/100**. El almacenamiento es mediante **tarjeta SD/MMC**. En la figura 52 puede verse la Raspberry Pi.


Figura 52: Raspberry Pi

La distribución Linux usada será **Raspbian**, una distribución basada en Debian ARM optimizada para el Raspberry Pi y recomendada por la Raspberry Pi Foundation. Raspbian se instalarán en una tarjeta SD de 8Gb. para realizar las pruebas.

Una vez que se tiene instalado Raspbian en la tarjeta SD y arancada en la Raspberry Pi es necesario instalar el demonio. El demonio sólo depende de un paquete que no viene instalado por defecto en Raspbian, que es **libusb-dev**. Una vez instalada se compila el demonio usando el **Makefile** como se indica en el *Anexo 1*. Además se instalará y configurará el arranque del demonio como se ha explicado en la *Implementación del Demonio*.

Para la prueba integrada serán necesarios los siguientes elementos hardware:

- **Raspberry Pi**.
- **Tarjeta SD** (con Raspbian y el demonio instalados).
- **Cable micro USB B macho-USB A macho** (para conectara la Raspberry Pi a la alimentación).
- **Periférico de monitorización y control**.
- **Cable USB B macho-USB A macho** (para conectar el periférico a la Raspberry).
- **Cable de red**.
- **Alimentador 220V-5V 1.5A USB**.


Figura 53: Elementos hardware necesarios

La conexión de estos elementos se realiza como puede observarse en la figura 54, conectando la Raspberry Pi al alimentador utilizado el cable micro USB-USB A. a su vez, el periférico se conectará a la Raspberry utilizando el cable USB B macho-USB A macho. Además, se conectará el cable de red, un extremo a la tarjeta de red de la Raspberry y el otro a una toma del

router.


Figura 54: Conexión de los elementos

Una vez que se tiene preparadas la conexiones, se conecta el alimentador a una toma de red eléctrica. La Raspberry Pi comenzará con el **arranque del sistema operativo**. Una vez que termine el arranque del sistema de la Raspberry (unos 20-30 segundos) ya puede utilizarse el periférico.

Con el sistema arrancado se procederá a la realización de las pruebas integradas, la primera parte de las pruebas consiste en la **comprobación de la correcta comunicación bidireccional** del periférico con el ordenador(la Raspberry Pi en este caso) y que todas las **funciones de monitorización y control operan correctamente**. A continuación se muestra el resultado de cada opción de monitorización y control con el resultado obtenido en la prueba.

Dentro del menú de ***Monitorización*** tenemos:

**Fecha y hora del sistema**

Se puede ver en la imagen como el periférico obtiene la hora y la fecha de las Raspberry Pi.

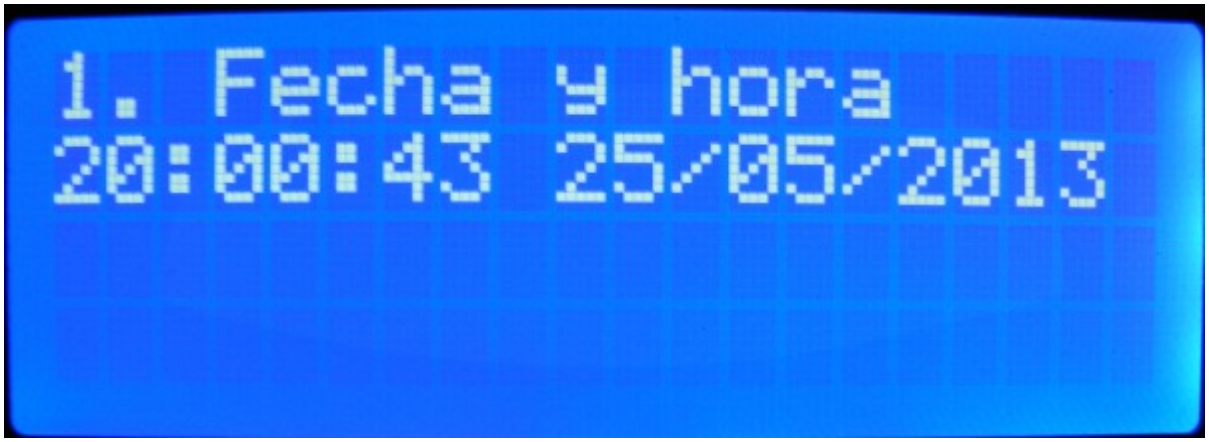


Figura 55: Prueba: Fecha y hora del sistema

**Espacio en disco**

En esta prueba se comprueba como se muestra en el periférico el tamaño de la partición raíz, el espacio ocupado, el espacio libre y el porcentaje de ocupación.

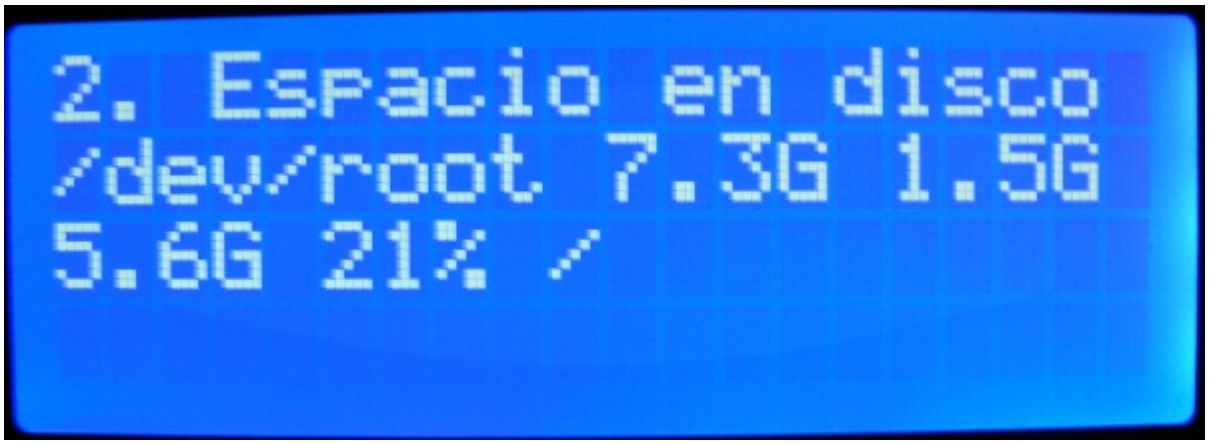


Figura 56: Prueba: Espacio en disco

**Uptime y carga del sistema**

Puede verse en la primera linea el tiempo que lleva arrancada la máquina, y en las siguientes la carga del sistema en los últimos 1, 5 y 15 minutos.

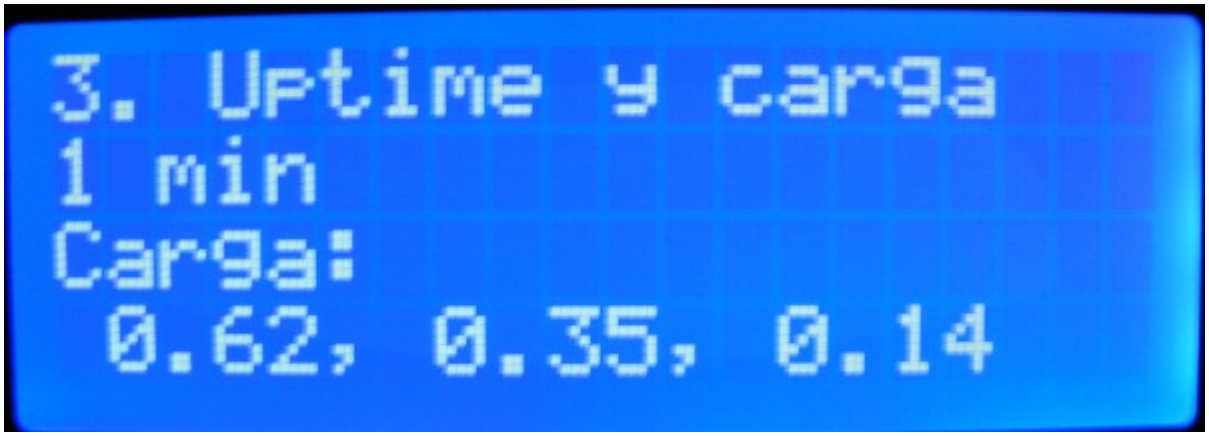


Figura 57: Prueba: Uptime y carga del sistema

**Memoria RAM libre**

Se muestra en la pantalla la memoria RAM total instalada en el sistema, la memoria usada y la memoria libre, todo expresado en Megabytes.


Figura 58: Prueba: Memoria RAM libre

**Kernel**
Se puede ver en la figura la versión del kernel y la arquitectura de las distribución Raspbian instalada en las raspberry.

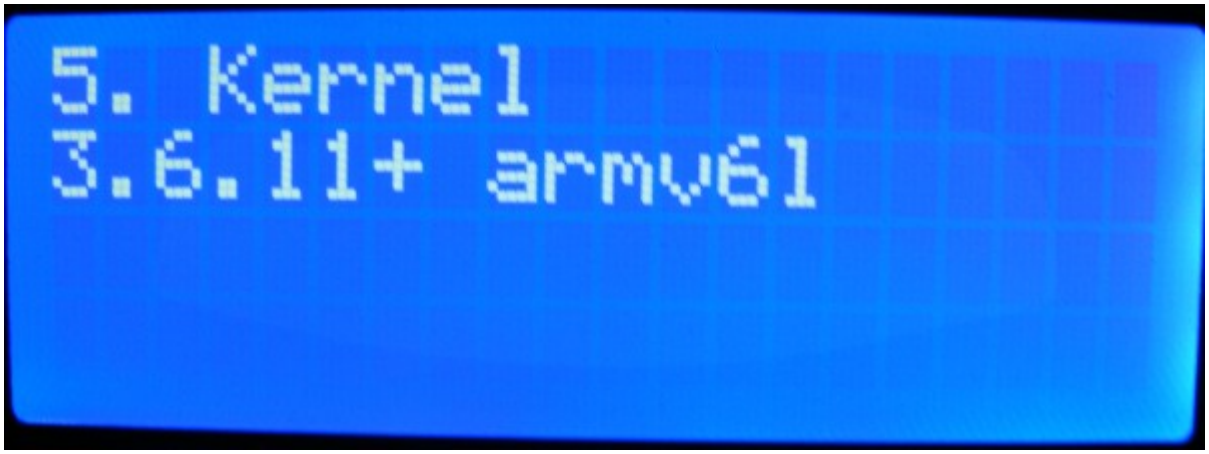


Figura 59: Prueba: Kernel

**Información de Red**
En la primera linea se obtiene la dirección MAC de la tarjeta de red y en la segunda la dirección IP asociada actualmente a esa tarjeta.

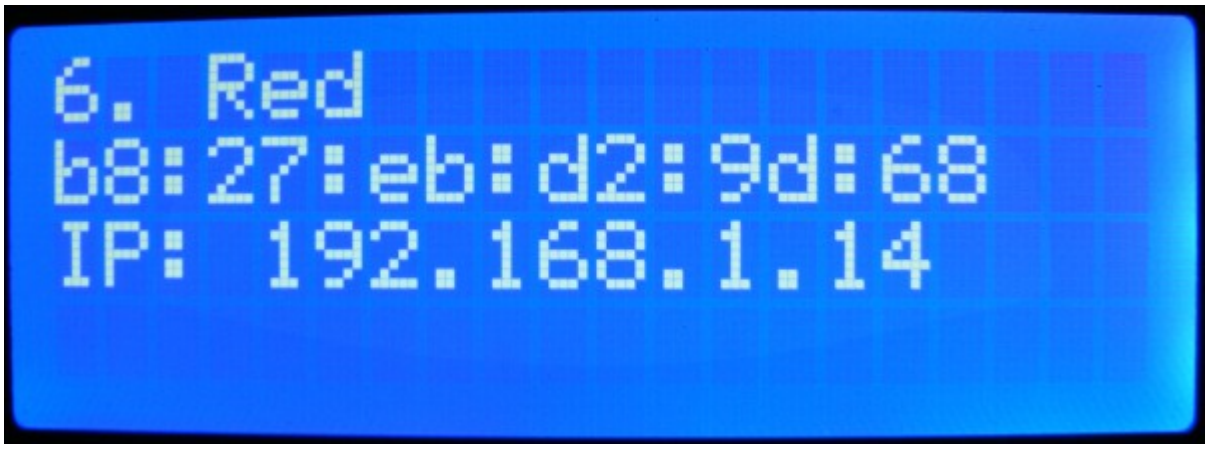


Figura 60: Prueba: Información de Red

**Temperatura de la CPU**
Se muestra correctamente la temperatura de la CPU de la Raspberry Pi.

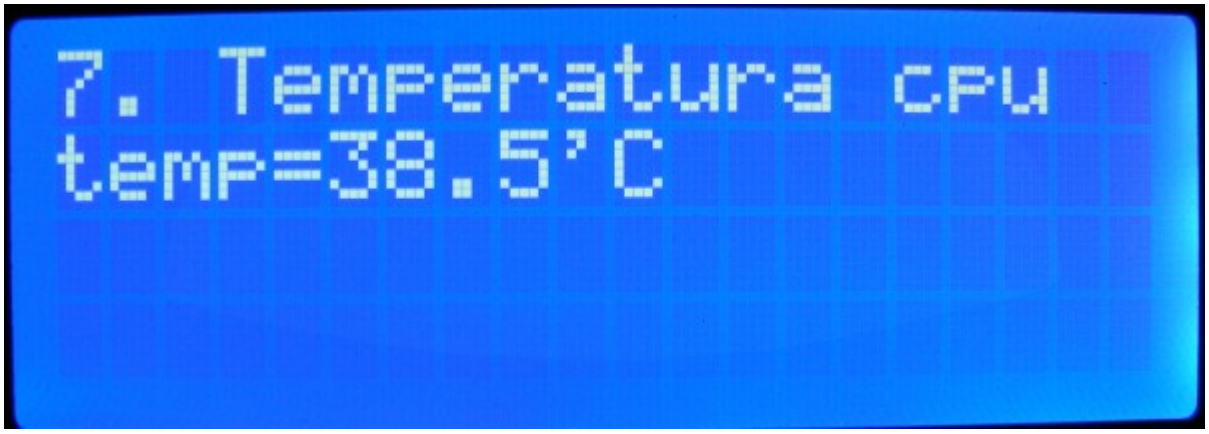


Figura 61: Prueba: Temperatura de la CPU

Dentro del menú de ***Control*** tenemos:

**Reiniciar equipo**
Cuando se pulsa el botón aceptar el demonio envía el mensaje al dispositivo que se observa en la imagen antes de reiniciarse.

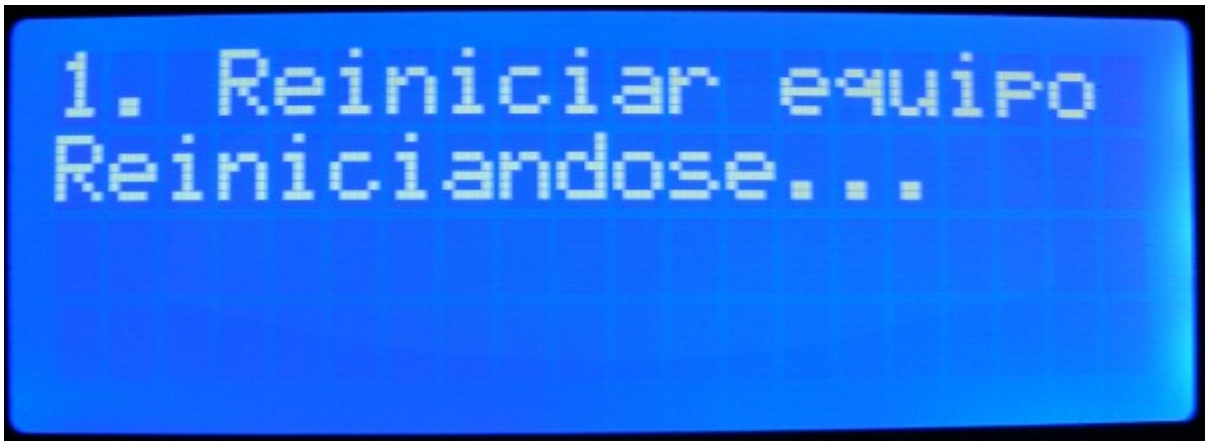


Figura 62: Prueba: Reiniciar equipo

**Apagar equipo**
Al igual que en el reinicio, en el apagado se envía el mensaje al periférico antes de comenzar el apagado del sistema.

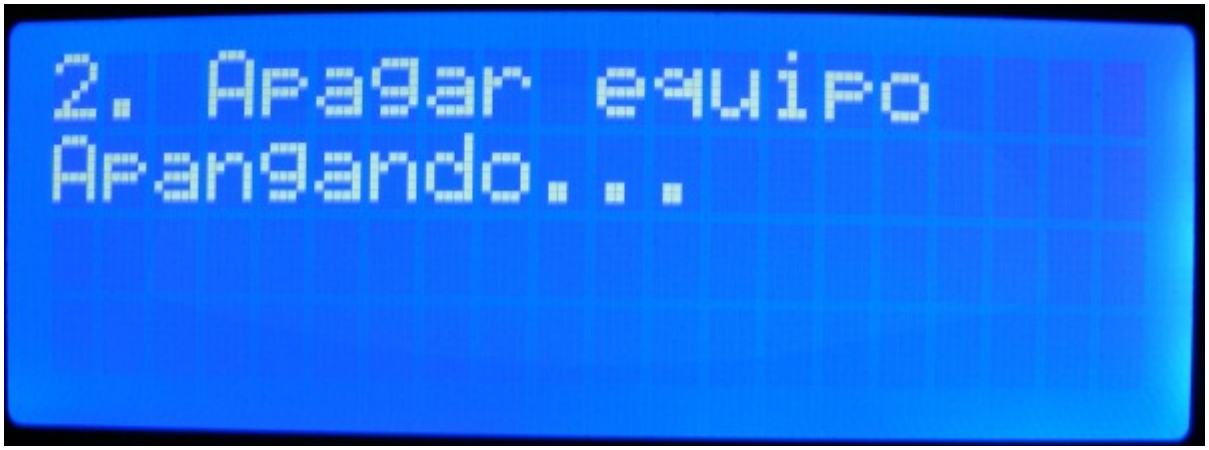


Figura 63: Prueba: Apagar equipo

**Pruebas de conexión-reconexión del periférico**

Para comprobar que el demonio responde correctamente a la **conexión y desconexión del periférico** del ordenador, se realiza esta prueba que se basa en una vez conectado el dispositivo y comprobar que funciona correctamente, se **desconecta del puerto USB**. Después de unos segundos se **vuelve a conectar el periférico** a la Raspberry Pi y se verifica que sigue funcionando todo. Se vuelve a repetir este proceso por segunda vez para validar que responde bien a varias conexiones-reconexiones. El demonio va escribiendo en su log (***/var/log/daemon.log***) cuando se conecta y desconecta el periférico, en el caso de las desconexiones se muestra el error correspondiente en caso que estuviera realizando alguna comunicación en el momento de la desconexión. El log se puede ver a continuación:

```
May 26 11:22:13 raspberrypi PMC[2171]: Demonio iniciado
May 26 11:22:13 raspberrypi PMC[2171]: Dispositivo conectado
May 26 11:22:47 raspberrypi PMC[2171]: Dispositivo desconectado:
    No error
May 26 11:23:11 raspberrypi PMC[2171]: Dispositivo conectado
May 26 11:23:41 raspberrypi PMC[2171]: Dispositivo desconectado:
    error sending control message: Broken pipe
May 26 11:23:47 raspberrypi PMC[2171]: Dispositivo conectado
```

**Pruebas de historificación de la temperatura**

Durante la realización de las pruebas anteriores el demonio ha ido historificando la temperatura de los sensores de temperatura en el fichero ***/var/log/PMChist.log***.

Para verificar el contenido del fichero, se conecta por SSH a la Rapberry PI y se puede comprobar que en el fichero se ha ido escribiendo la temperatura cada 30 segundos. Se puede observar a continuación un fragmento de este fichero:

```
25-05-2013 18:29:05,025
25-05-2013 18:29:35,026
25-05-2013 18:30:05,026
25-05-2013 18:30:35,026
25-05-2013 18:31:05,027
```

## 6.3. Pruebas de consumo

Debido a que el periférico debe consumir menos de 500mA para que se pueda alimentar desde el puerto USB del ordenador, se verificará en este apartado que eso se está cumpliendo, realizando unas medidas de consumo del periférico.

Para realizar esta medidas, se ha fabricado un **latiguillo USB**, con un conector USB hembra y un conector USB macho, como se puede ver en la figura 64, dejando suelto los extremos del cable de 5V pelados. Esto es así para poder **medir la intensidad** de corriente con un **polímetro**, ya que se debe colocar **en serie** al circuito.

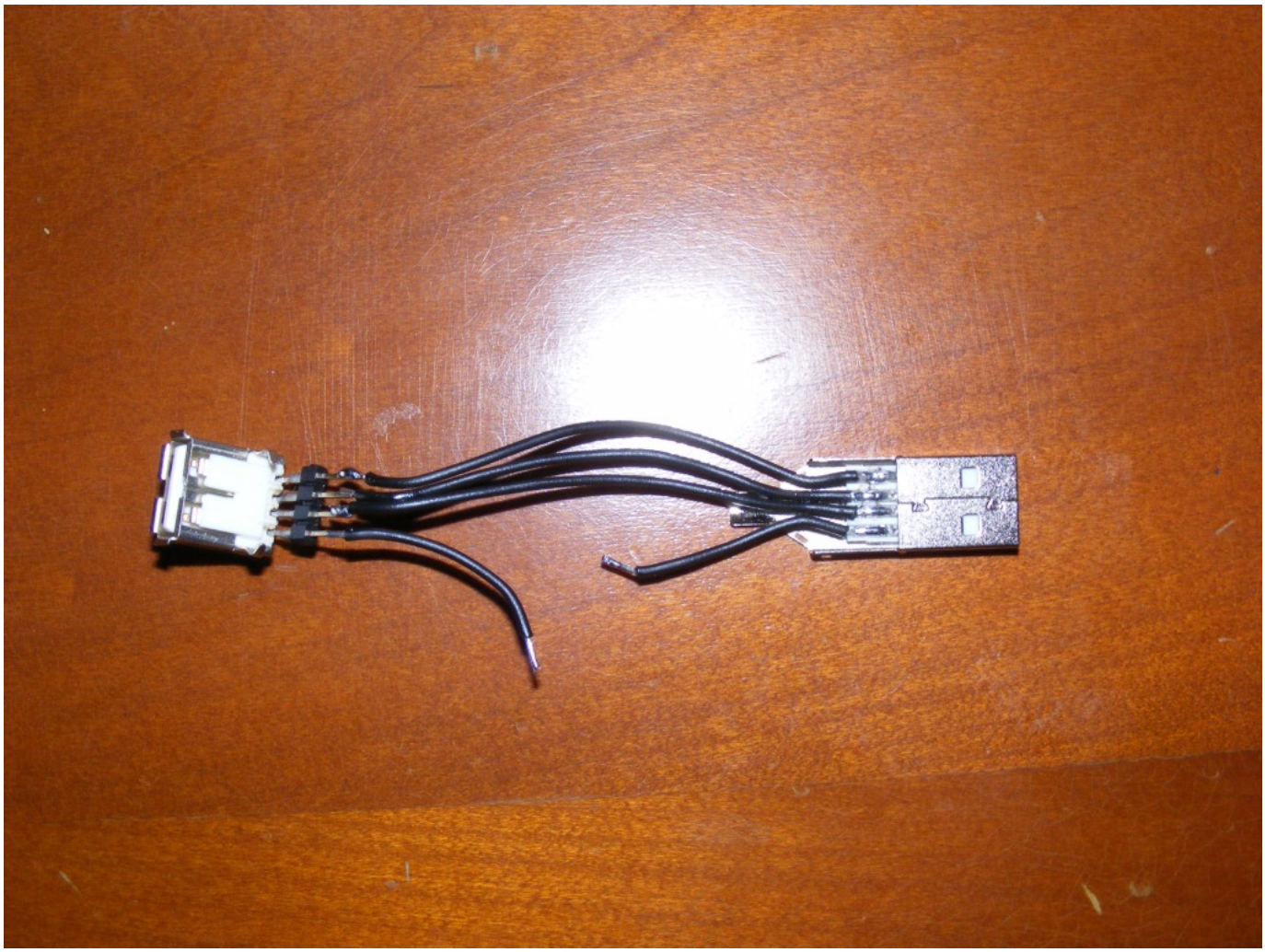

Figura 64: Latiguillo USB

El polímetro se coloca en la posición para medir intensidades en corriente continua, y se conecta a los dos cables pelados. Una vez hecho esto, se conecta el periférico al conector USB hembra del latiguillo y el conector USB macho del latiguillo se conecta a la Raspberry Pi.

Se puede ver en la figura 65 la medida de intensidad tomada por el polímetro expresada en miliamperios (mA).

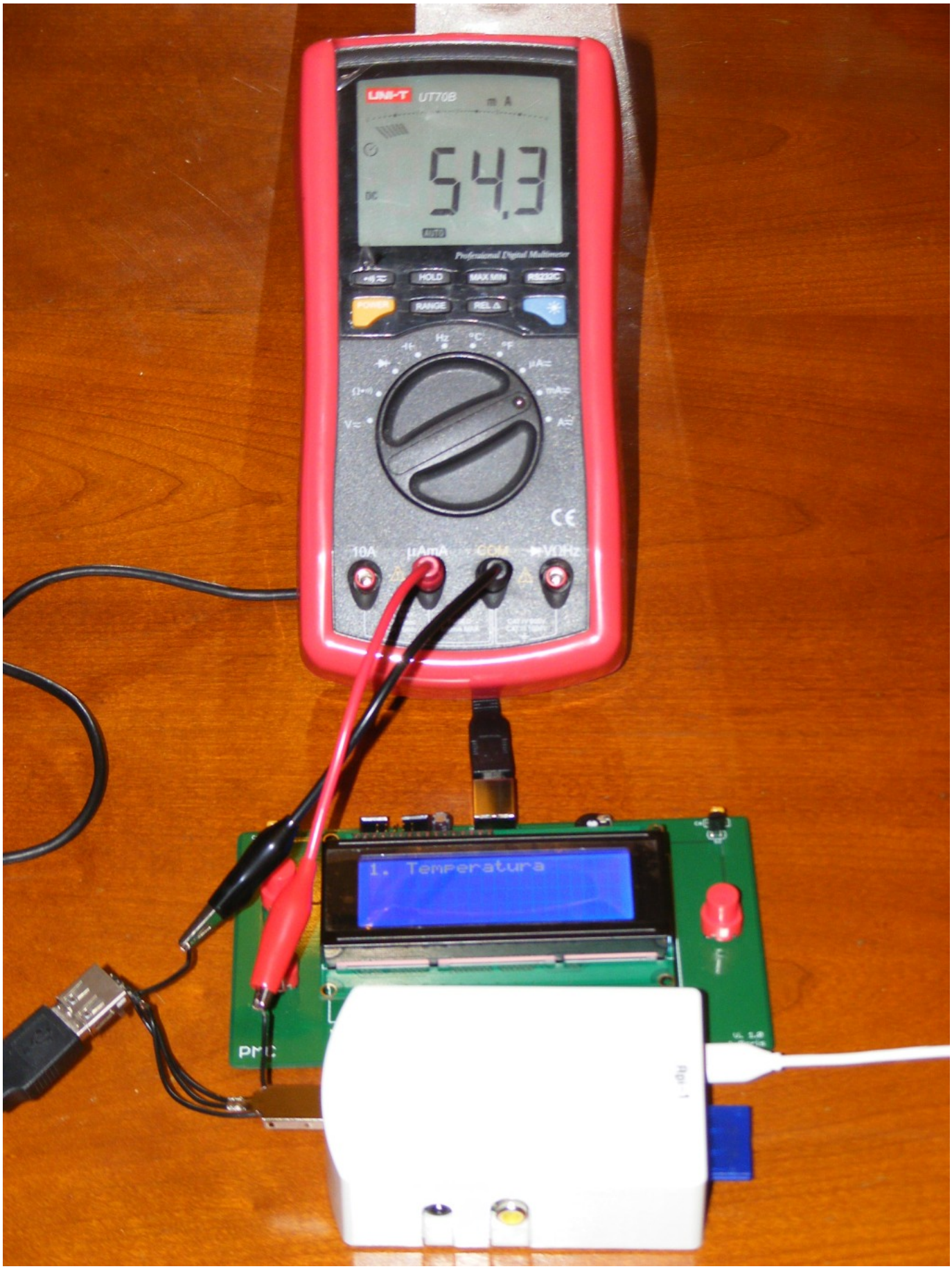


Figura 65: Latiguillo USB

La medida obtenida es de **54.3 mA** por lo que se encuentra por debajo de los 500mA máximos recomendados por la especificación del protocolo USB por cada puerto. Por otro lado, es interesante calcular la **potencia eléctrica** que consumirá el dispositivo, expresada en vatios(W). La formula para el calculo de la potencia eléctrica en corriente continua es la siguiente:

$$P = I \cdot V$$

Dónde $P$ será la potencia eléctrica calculada expresada en vatios, $I$ es la intensidad de corriente expresada en amperios y $V$ es voltaje expresado en voltios. El voltaje del puerto USB son 5V, por lo que podemos despejar la formula de la siguiente manera:

$$P = 0{,}0543 \cdot 5 = 0{,}2715W$$

Por tanto el periférico de monitorización y control consume **0,2715 W**, un consumo muy reducido.

# 7. Consideraciones finales

En esta sección se presentará la planificación del proyecto, el presupuesto, además de las conclusiones y trabajos futuros.

## 7.1. Planificación

En la figura 66 se muestra el diagrama de Gantt de la planificación del presente proyecto.

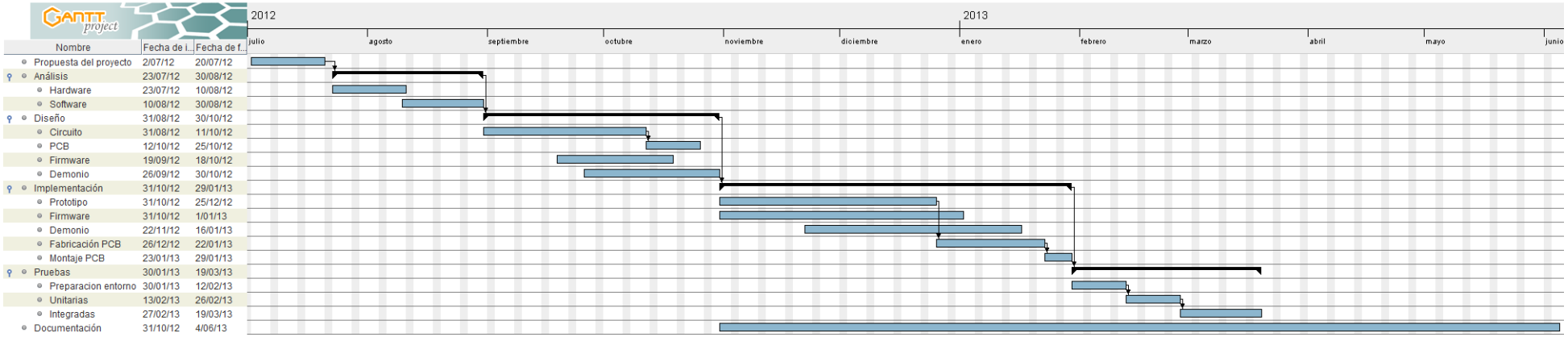


Figura 66: Planificación

## 7.2. Presupuesto

En esta sección se detalla el presupuesto desglosado de este proyecto.

**Costes de Personal**

En el presente proyecto únicamente ha participado un ingeniero informático durante los 11 meses de duración del mismo. Se han trabajado **232 días hábiles**, como se puede ver en la planificación, estimando un trabajo de **4 horas por día** laborable aproximadamente, de los cuales 54 días corresponden a un ingeniero hardware, 5 días a un técnico de montaje y los restantes 173 a un ingeniero software.

El coste del ingeniero software serían 60€/hora, el del ingeniero hardware 70€/hora y el del técnico de montaje 30€/hora, por consiguiente los costes totales de personal se muestran en la siguiente tabla:

| **Concepto** | **Salario hora** | **Horas** | **Total** |
|---|---|---|---|
| Ingeniero Software | 60,00€ | 692 horas | 41.520,00€ |
| Ingeniero Hardware | 70,00€ | 216 horas | 15.120,00€ |
| Técnico de montaje | 30,00€ | 20 horas | 600,00€ |
| | | **Total** | 57.240,00€ |

Tabla 39: Costes de Personal

**Costes de Equipo**

En ese apartado se detallan los costes del equipo utilizado para el proyecto, incluyendo el ordenador portátil comprado al comienzo del proyecto.

| **Concepto** | **Precio** |
|---|---|
| Portatil Sony Vaio SV-E1711C5E | 682,62€ |
| Raspberry Pi Tipo B | 30,00€ |
| **Total** | 712,62€ |

Tabla 40: Costes de Equipo

**Costes de Hardware**
Se procede a detallar el coste de los materiales para construir el periférico, sin incluir las herramientas necesarias ya que se disponía de ellas antes del comienzo del proyecto.

| Cantidad | Concepto | Precio unitario | Precio total |
|---|---|---|---|
| 5 | Placa PCB | 7,44€ | 37,20€ |
| 1 | ATMEGA168A-PU | 2,50€ | 2,50€ |
| 1 | Pantalla LCD 20X4 | 12,70€ | 12,70€ |
| 1 | Zócalo de 28 pines torneado | 0,78€ | 0,78€ |
| 1 | Cristal cuarzo 12Mhz. | 1,38€ | 1,38€ |
| 3 | Sensor de temperatura LM35 | 2,96€ | 8,88€ |
| 3 | Pulsadores PCB | 0,70€ | 2,10€ |
| 1 | Conector USB hembra tipo B PCB | 1,00€ | 1,00€ |
| 1 | Placa de prototipos (breadboard) | 42,55€ | 42,55€ |
| 1 | Resistencias, condensadores y diodos | 2,50€ | 2,50€ |
| | | **Total** | 111,59€ |

Tabla 41: Costes de Hardware

**Otros costes**

En este apartado se incluyen otros costes derivados, como los gastos de aduana de las PCB debido que venían desde China.

| Concepto | Precio |
|---|---|
| Gastos de aduana de las PCB | 34,00€ |
| **Total** | 34,00 € |

Tabla 42: Otros costes

### Presupuesto total

A continuación se calcula el total de los costes, añadiendo el riesgo, beneficio y los impuestos.

| **Concepto** | **Total** |
|---|---|
| Costes de personal | 57.240,00€ |
| Costes de equipo | 712,62€ |
| Costes de hardware | 111,59€ |
| Otros costes | 34,00€ |
| **Subtotal** | 58.098,21€ |
| Riesgo (10 %) | 5.809,82€ |
| Beneficio (33 %) | 19.172,40€ |
| **Total sin IVA** | 83.080,43 € |
| IVA (21 %) | 17.446,89€ |
| **Total** | 100.527,32€ |

Tabla 43: Presupuesto final

El presupuesto total de ejecución de este proyecto asciende a la cantidad de **OCHENTA Y TRES MIL OCHENTA CON CUARENTA Y TRES EUROS** sin impuesto del valor añadido.

Incluyendo el 21 % de IVA el presupuesto final es de **CIEN MIL QUINIENTOS VEINTISIETE CON TREINTA Y DOS EUROS**.

## 7.3. Conclusiones

Una de las conclusiones principales de este proyecto es que se ha conseguido desarrollar un **periférico** que se conecta por el **puerto USB** desde cero, tanto el **hardware** como el **software**.El periférico se encarga de realizar tareas de monitorización y control del ordenador al que está conectado por USB.

La idea inicial de este proyecto era realizar un **ordenador basado en un microcontrolador** desde cero, pero se decidió realizar un periférico en lugar de un ordenador debido a que la complejidad que tenia y el tiempo necesario para llevarlo a cabo se quedan fuera de lo que se pretende con un proyecto de fin de carrera. Finalmente se decidió realizar un proyecto sobre un periférico basado en un microcontrolador y que se conectara por puerto USB.

Durante el desarrollo de este proyecto se han ido adquiriendo conocimientos sobre la programación de microcontroladores en **AVR-C**, que permite una programación tanto de alto nivel como de bajo nivel, y existe una amplia documentación y soporte de la comunidad, que ha permitido resolver cualquier duda y problema, además de contar con muchas librerías para periféricos como la **pantalla LCD**. Además se ha descubierto el manejo de dispositivos analógicos, en concreto el **sensor de temperatura LM35**, y los problemas que se pueden encontrar si fallase, por lo que se ha implementado un sistema de redundancia hardware.

Por otro lado, gracias a la **librería V-USB**, la comunicación del periférico con el ordenador ha sido bastante sencilla y no se tuvieron problemas para realizar una comunicación bidireccional por USB. Aunque inicialmente hubo problemas para realizar una comunicación USB, debido a que se usó una cable que no cumplía el estándar USB(era una simple manguera de 4 hilos por 0.25mm). Luego después de cambiarlo por un cable USB estándar se solucionaron los problemas.

Otros problemas surgidos durante la realización del proyecto, fue que se quemó el microcontrolador comprado originalmente, un Atmega328, y se sustituyó por el usado finalmente, el Atmega168.

Por otro lado, durante la fase de implementación se cambió la **pantalla LCD** original (de **16x2** caracteres) por la de **20x4** caracteres debido a que con una sola línea para la información resultaba insuficiente. Además gracias que el pineado de la pantalla es igual, solo supuso cambiar algunas configuraciones en el código del firmware para aprovechar todas las líneas de la nueva pantalla.

Tras la fase de pruebas, se ha comprobado que el **periférico cumple los objetivos y funcionalidades** planteadas y se ha visto plasmadas las ilusiones iniciales en un periférico completo y funcional, tanto la parta hardware y la parte software.

Finalmente, cabe destacar que el periférico sería expandible pudiendo añadirse más tipos de datos monitorizados, con más menús, ya que de los 16kb de memoria flash del microcontrolador solo se han consumido 5kb.

## 7.4. Trabajos futuros

Pese a que este proyecto de fin de carrera ha cubierto sus objetivos, el periférico tiene muchas lineas para ser mejorado, ampliar su capacidades y mejorar los puntos en que existan carencias. En el presente apartado se comentarán algunos puntos posibles para un trabajo futuro sobre este proyecto:

- **Pantalla LCD gráfica**
  Para poder monitorizar más datos, presentando mayor información y con mas detalle, e incluso mostrando gráficas(como por ejemplo un gráfico histórico de la temperatura, la carga, etc.), se podría cambiar la pantalla LCD orientada a caracteres por una **pantalla LCD gráfica**. Una pantalla gráfica posible sería la que se muestra en la imagen 67, que se trata de una pantalla LCD gráfica de 128x64 pixeles.

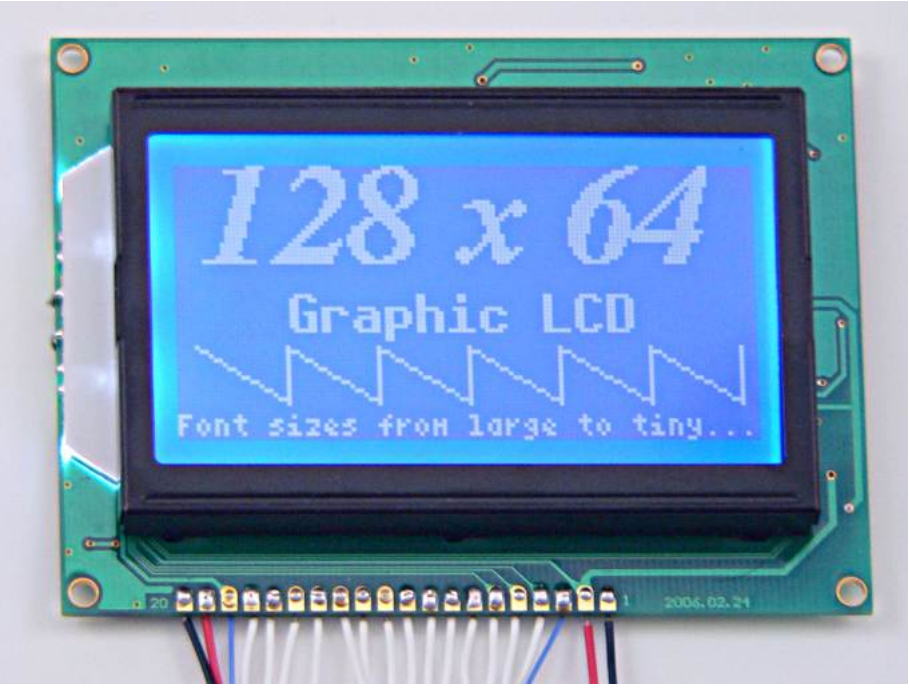


Figura 67: Pantalla LCD 128x64

  No se ha utilizado en este proyecto (aunque se pretendría) porque la pantalla que se compró se dañó el modo serial, y en modo paralelo no había suficiente pines de entrada salida del microcontrolador.

- **Demonio para Windows**
  Como un trabajo futuro se podría considerar adaptar el demonio a un **servicio de Windows**. La librería libusb existe en Windows por lo que sólo sería adaptarlo para registrarlo como servicio de Windows y los comandos para acceder a los datos monitorizados. No se ha realizado en el presente proyecto porque tras en análisis se comprobó que el sistema operativo más utilizado en servidores era los Unix-like.

- **Conectividad inalámbrica**
  Actualmente se realiza la comunicación entre el periférico y el ordenador por USB, pero se podría plantear realizar la comunicación de manera inalámbrica, por ejemplo mediante **bluetooth**. En el mercado existen módulo bluetooth con comunicación RS232, por lo que la comunicación con el microcontrolador utilizado sería factible. En la siguiente figura se muestra uno de estos módulos bluetooth.


Figura 68: Módulo bluetooth

- **Periférico en caja**
  Para evitar que queden los componentes al aire y dar un mejor aspecto, se podría meter el periférico en un caja de plástico, mecanizada para para la pantalla, conectores y pulsadores. No se ha metido en una caja en este proyecto ya que no se encontró una caja adecuada y manejable. En la siguiente figura se muestra el proyecto de la asignatura Sistemas Informáticos que sí se montó en una caja.


Figura 69: Proyecto Sistemas Informáticos

- **Driver en lugar de un demonio**
  Como trabajo futuro del proyecto podría cambiar el demonio implementado por un driver, que se cargue cuando el dispositivo sea conectado al puerto USB, así se evitaría tener que comprobar cada segundo si el periférico está conectado. No se ha realizado en este proyecto por la complejidad y la falta de documentación para interactuar con el puerto usb a bajo nivel.

# 8. Acrónimos y definiciones

**ADC** Analog-to-Digital Converter (Conversor analógico-digital).

**ARM** Advanced RISC Machine.

**ASCII** American Standard Code for Information Interchange (Código Estándar Estadounidense para el Intercambio de Información).

**BASIC** Beginner's All-purpose Symbolic Instruction Code (Código simbólico de instrucciones de propósito general para principiantes).

**CD** Compact disc (Disco compacto).

**CMOS** Complementary metal-oxide-semiconductor.

**CPU** Central Processing Unit (Unidad central de procesamiento).

**CSV** Comma-Separated Values (Valores separados por comas).

**DIL** Dual in-line.

**EEPROM** Electrically Erasable Programmable Read-Only Memory (ROM programable y borrable eléctricamente).

**EPROM** Erasable Programmable Read-Only Memory (ROM programable borrable).

**ESA** European Space Agency (Agencia espacial Europea).

**GCC** GNU Compiler Collection.

**GND** Ground (Masa).

**GNU** GNU's not Unix.

**GPU** Graphics processing unit (Unidad de procesamiento gráfico).

**HDMI** High-Definition Multimedia Interface.

**IP** Internet Protocol.

**LCD** Liquid Crystal Display (Pantalla de cristal líquido).

**MAC** Media Access Control.

**MIPS** Million instructions per second (Millones de instrucciones por segundo).

**MMC** MMultiMediaCard.

**OTP ROM** One Time Programable Read Only Memory (Memoria de sólo lectura programable una vez.

**PC** Personal Computer (Ordenador personal).

**PCB** Printed Circuit Board) (placa de circuito impreso).

**PFC** Proyecto fin de carrera.

**PID** Process ID (Identificador de proceso).

**RAM** Random-Access Memory (memoria de acceso aleatorio).

**RISC** Reduced Instruction Set Computer.

**ROM** Read Only Memory (Memoria de sólo lectura).

**SBC** Single Board Computer (Ordenador en una sola placa).

**SD** Secure Digital.

**SSH** Secure SHell (interprete de comandos seguro).

**TTY** Teletype (Teletipo).

**USB** Universal Serial Bus (Bus de serie universal).

**Breadboard** O placa de prototipos es un tablero con orificios conectado eléctricamente entre si siguiendo patrones de lineas, preparados para insertar componentes y cables.

**Diodo zener** Es un diodo de de cromo que se ha construido para que funcione en las zonas de rupturas y recibe ese nombre por su inventor.

**Enlace simbólico** En sistemas unix, indica un acceso a un directorio o fichero que se encuentra en un lugar distinto dentro de la estructura de directorios. Una modificación realizada utilizando este enlace se reflejará en el original; pero, por el contrario, si se elimina el enlace, no se eliminará el auténtico.

**Firmware** Bloque de instrucciones de máquina para propósitos específicos, grabado en una memoria de tipo de solo lectura, que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo de cualquier tipo.

**Hot plug** (Conexión en caliente). Capacidad de un dispositivo de conectarse al ordenador sin necesidad de apagarlo previamente.

**Makefile** Es un fichero de texto que utiliza el programa make para llevar la gestión de la compilación de programas.

**Plug and play** (Enchufar y usar) Tecnología que permite usar un dispositivo en un ordenador sin tener que configurarlo.

**Pulsador normalmente abierto** Pulsador que en su estado de reposo, deja el circuito abierto y cuando es pulsado, cierra el circuito.

**Resistencia de Pullup** Es una resistencia conectada a alimentación positiva que sirve para elevar la tensión de salida de un circuito lógico.

**Runlevel** O nivel de ejecución se refiere al modo de operación en los sistemas operativos que implementan el estilo de sistema de arranque de iniciación tipo UNIX System V. El runlevel 0 es apagado y el 6 reiniciado. Los intermedios difieren en relación a qué unidades de disco se montan, y qué servicios de red son iniciados.

**Sistema empotrado** Sistema diseñado para realizar algunas pocas funciones especificas.

# Referencias

[1] http://w3techs.com/technologies/overview/operating_system/all

[2] http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/ESA/BSSC962.PDF

[3] http://www.cadsoftusa.com

[4] http://www.seeedstudio.com/depot/fusion-pcb-service-p-835.html

[5] http://www.nongnu.org/avr-libc/

[6] http://www.libusb.org

[7] http://www.raspbian.org/

[8] http://www.obdev.at/products/vusb/index.html

[9] http://homepage.hispeed.ch/peterfleury/avr-lcd44780.html

[10] http://es.wikipedia.org

[11] http://codeandlife.com/2012/01/22/avr-attiny-usb-tutorial-part-1/

[12] http://www.atmel.com/devices/atmega168.aspx

[13] http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en020415

[14] http://www.fischl.de/usbasp/

[15] http://www.teknoplof.com/2010/08/03/no-te-pierdas-con-los-conectores-usb/

[16] http://www.netzmafia.de/skripten/unix/linux-daemon-howto.html

# A. Anexo 1: Compilación del código y programación del microcontrolador

En este anexo se procederá a explicar cómo **compilar el código fuente** (tanto el del firmware del microcontrolador como el del demonio UNIX), además de explicarse como **programar el microcontrolador** (es decir, grabar el firmware compilado en el microcontrolador).

### Compilación del código del demonio

El código del demonio incluido en el CD adjunto se compone de los siguientes ficheros:

- **PMC.c**: El código del demonio.
- **Makefile**: El makefile para compilar el código e instalarlo.
- **init.d/PMC**: El script de arranque del demonio.

Para poder poder compilarlo es necesario tener instalado el paquete **libusb-dev**(en distribuciones Linux Debian y derivadas) además de las librerías estándar de desarrollo de **libc** junto al compilador **GCC**. Para compilar el demonio e instalarlo se hace utilizando el makefile, con los siguientes comandos:

```
make clean
make
sudo make install
```

El primer comando limpia cualquier compilación anterior, el segundo compila el código fuente y el tercero instala el demonio en ***/usr/local/bin*** además de instalar el script de arranque del demonio en***/etc/init.D*** .

### Compilación del código del firmware

Ahora es necesario compilar otra pieza fundamental del software de ese proyecto, el firmware del microcontrolador. Para poder compilar el firmware son necesarios los siguientes paquetes (en distribuciones Linux Debian y derivadas):

- **gcc-avr**
- **binutils-avr**

Los ficheros incluidos en el CD en la carpeta del código del firmware del microcontrolador son los siguientes:

```
usbdrv(directorio)
lcd.c
lcd.h
main.c
Makefile
usbconfig.h
```

Se incluyen todos los ficheros de código fuente creados así como el código de las librerías para tratar la pantalla LCD y la librería V-USB, y el makefile para compilar todo.

Para poder compilar el código se utiliza el makefile, usando los siguiente comandos:

```
make clean
make hex
```

Se genera un **fichero .hex** que es el volcado binario del código compilado listo para poder grabarlo en el microcontrolador.

### Programación del firmware en el microcontrolador

Una vez que se tiene el fichero compilado hex del firmware se puede programar en el microcontrolador. Para poder grabar el firmware en el microcontrolador es necesario tener un programador lo transferirá desde el ordenador. Existen multitud de programadores, y se ha elegido uno basado en **USBasp**[14], que es un diseño libre de un programador de microcontroladores de la familia AVR de Atmel que se conectar por puerto USB. En la figura 70 puede verse el programador adquirido con su cable de conexión.

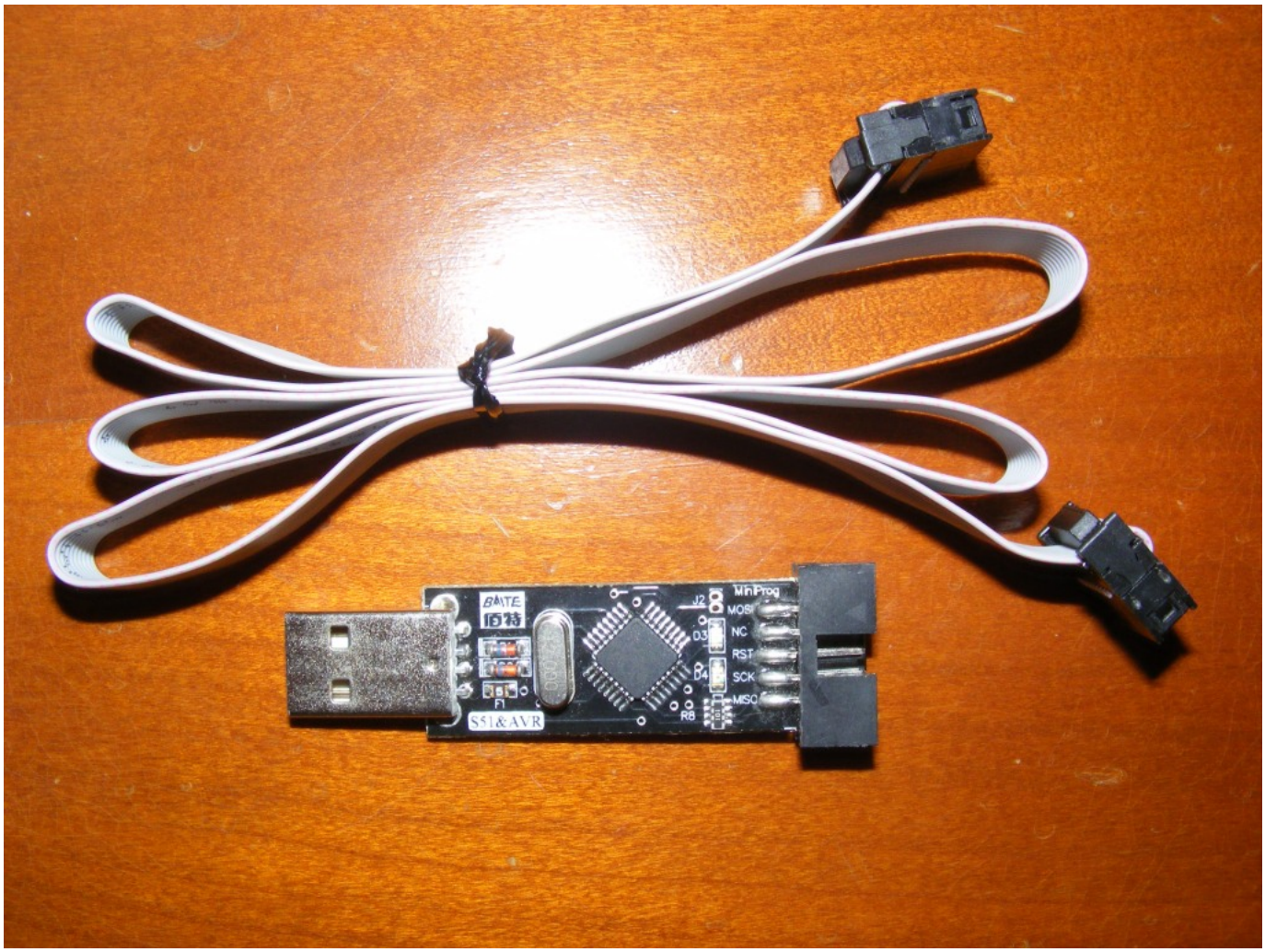


Figura 70: Programador USBasp

Dicho programador se conecta al ordenador por puerto **USB** y al microcontrolador por el puerto **AVR-SPI**, que cuyo conector macho está incluido en la placa del periférico, por lo que se puede conectar directamente el programador como se muestra en la figura 71.

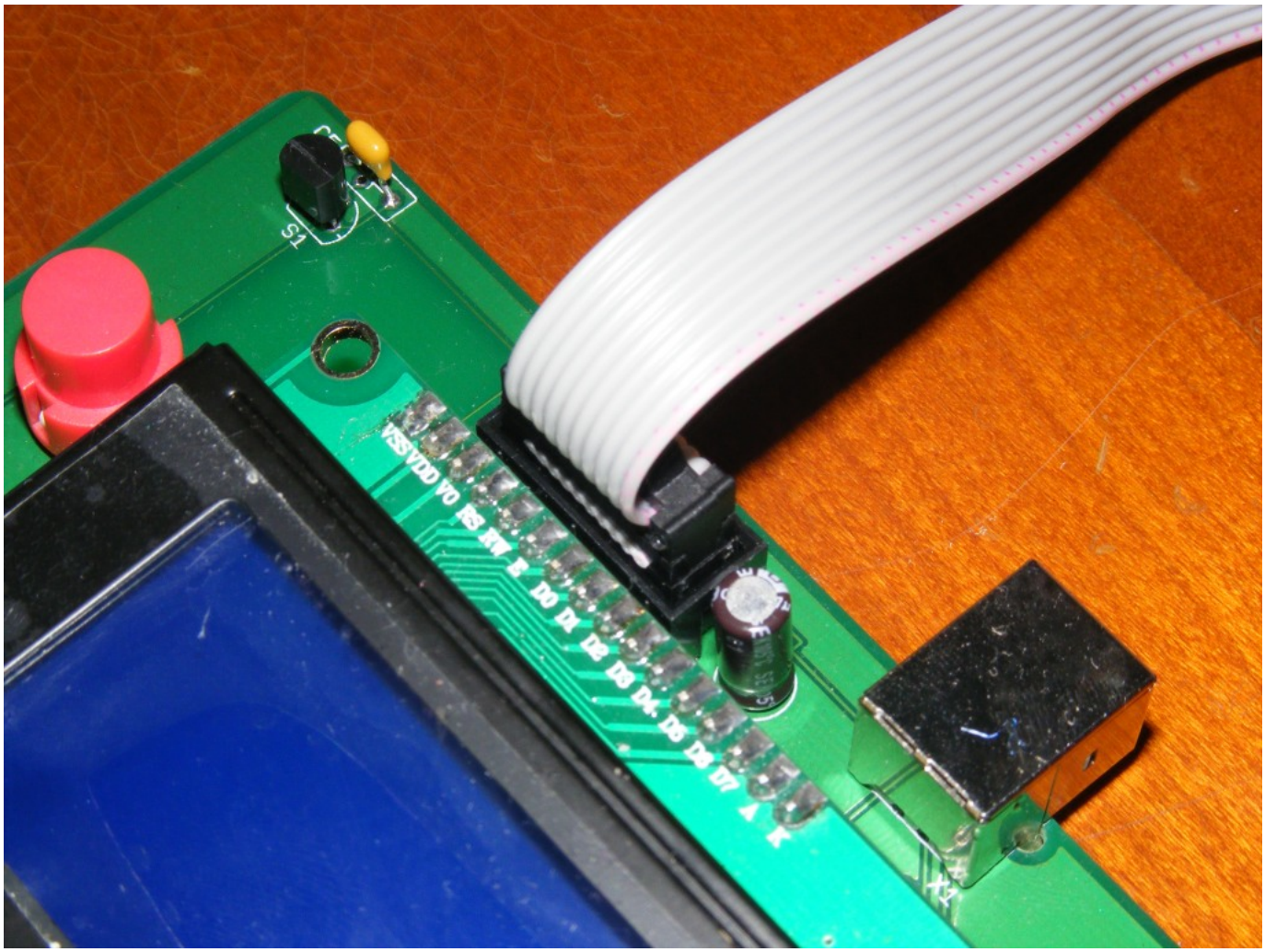


Figura 71: Conexión del periférico con el programador

Para poder grabar el firmware en el microcontrolador es necesario el siguiente paquete:

- **avrdude**

Una vez que se tiene conectado periférico al programador y el programador por USB al ordenador se puede proceder a grabar el firmware en el microcontrolador, usando el Makefile con el siguiente comando:

```
sudo make program
```

Este comando establece los bytes de configuración del microcontrolador (que indican el tipo de cristal utilizado, si está activado el watchdog, etc.) y graba el firmware.